Seguimiento y búsqueda de objetivos en entornos complejos usando micro vehículos aéreos con cámaras monoculares para aplicaciones militares

Chauca Vera, Bryan Andrés

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización Y Control

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

PhD. Aguilar Castillo, Wilbert Geovanny

27 de agosto del 2020

Urkund Analysis Result

Analysed Document: TESIS_CHAUCA_BRYAN_urkund.pdf (D78121896)
Submitted: 8/24/2020 4:25:00 PM
Submitted By: bachauca@espe.edu.ec
Significance: 1 %

Sources included in the report:

Tesis-Quisaguano.pdf (D44942758)
tesis_grijalva_santiago.pdf (D57099779)
Tesis-Alvarez.pdf (D44942716)
https://repositorio.uam.es/bitstream/handle/10486/688901/barrio_algarabel_alicia_tfg.pdf?sequence=1

Instances where selected sources appear:

5

Dr. Wilbert G Aguilar
Director de Tesis

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación "**Seguimiento y búsqueda de objetivos en entornos complejos usando micro vehículos aéreos con cámaras monoculares para aplicaciones militares**" fue realizado por el señor **Chauca Vera, Bryan Andrés,** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito autorizar para que lo sustente públicamente.

Sangolquí, 27 de agosto de 2020

Firma:

PhD. Aguilar Castillo, Wilbert Geovanny

C.C: 0703844696

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

RESPONSABILIDAD DE AUTORÍA

Yo, **Chauca Vera, Bryan Andrés,** con cédula de ciudadanía n° 1803758406, declaro que el contenido, ideas y criterios del trabajo de titulación: "**Seguimiento y búsqueda de objetivos en entornos complejos usando micro vehículos aéreos con cámaras monoculares para aplicaciones militares"** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 27 de agosto de 2020

Firma:

Chauca Vera, Bryan Andrés

C.C: 1803758406

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN DE PUBLICACIÓN

Yo, **Chauca Vera, Bryan Andrés,** con cédula de ciudadanía n° 1803758406, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "**Seguimiento y búsqueda de objetivos en entornos complejos usando micro vehículos aéreos con cámaras monoculares para aplicaciones militares"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 27 de agosto de 2020

Firma:

Chauca Vera, Bryan Andrés

C.C: 1803758406

Dedicatoria

Este trabajo es dedicado a quienes más admiro, mis padres, que me han acompañado durante todo este trayecto, siendo esa inspiración para lograr mis objetivos.

También lo dedico a todas aquellas personas entusiastas de la tecnología, que, con optimismo, perseverancia y trabajo duro, día, a día aportan al desarrollo tecnológico del país, motivando a las nuevas generaciones a buscar y construir conocimiento a través de buenas prácticas.

Bryan Andrés Chauca Vera

Agradecimiento

Agradezco primeramente a Dios, que día tras día nos ofrece el milagro de vivir y estar junto a nuestros seres queridos compartiendo y disfrutando de su amor y sus bondades.

A mis padres que con su sacrificio y humildad sacaron adelante nuestra familia. Me educaron, me brindaron la oportunidad de prepararme y gracias a ellos ahora para mí la vida es un poco más cómoda y más feliz, que faltan las palabras en agradecimiento.

A mis hermanos, a mi familia, amigos y profesores que a lo largo de mi vida a portaron en mi crecimiento personal y profesional.

Bryan Andrés Chauca Vera

Índice de contenidos

Índice de tablas

Índice de figuras

Resumen

El presente proyecto de investigación consiste en el uso de AI para la implementación de un sistema de detección, seguimiento y búsqueda de un objetivo (cualquier persona), mediante el empleo de un micro-UAV Parrot Bebop 2 y de las imágenes proporcionadas por su cámara a bordo. Para la detección de personas, se usó el modelo de aprendizaje profundo YOLOv4, muy preciso y rápido para realizar esta tarea. A ello se adapta un sistema de seguimiento y rastreo digital basado en un Filtro de Kalman que integra una etapa de asignación de Id's a cada persona detectada. En este punto es necesaria la intervención de un usuario, quien, mediante un clic sobre cualquier persona en la imagen, indicará al sistema a quien seguir. Este seguimiento físico, es posible gracias a un controlador PI en pitch y yaw diseñados en función del modelo matemático obtenido de la estimación del movimiento del micro-UAV. Para potenciar su autonomía se incluye un modelo no supervisado de estimación de profundidad monodepth, que gracias a un mapa de profundidad propicia la detección y evasión de obstáculos. Además, al sistema se integra una etapa de búsqueda y re-identificación de persona a través de un modelo de CNN entrenada con las imágenes de la persona objetivo. El rendimiento del sistema es evaluado en un escenario interno controlado y un externo parcialmente controlado.

PALABRAS CLAVE:

- **NAVEGACION AUTONOMA**
- **DETECCION DE OBJETOS**
- **SEGUIMIENTO Y RASTREO DE PERSONAS**
- **EVASION DE OBSTACULOS**
- **BUSQUEDA Y RE-IDENTIFIACION DE PERSONAS**

Abstract

The present research project consists of using AI for the implementation of a detection, tracking and search system for a target (anyone), using a micro-UAV Parrot Bebop 2 and the images provided by their camera to board. For the person detection, was used the YOLOv4 deep learning model, very accurate and fast to carry out this task. A digital monitoring and tracking system based on a Kalman Filter is adapted to this, integrating a stage of assigning Id's to each detected person. At this point, the intervention of a user is necessary, who, by clicking on any person in the image, will indicate to the system who to follow. This physical tracking is possible thanks to a PI controller in pitch and yaw designed according to the mathematical model obtained from the estimation of the movement of the micro-UAV. To enhance its autonomy, an unsupervised depth estimation model is included, which thanks to a depth map favors the detection and obstacle avoidance. In addition, the system integrates a person search and re-identification stage through a CNN model trained with the images of the target person. The performance of the system is evaluated in a controlled internal scenario and a partially controlled external scenario.

KEYWORDS:

- **AUTONOMOUS NAVIGATION**
- **OBJECT DETECTION**
- **PERSON TRACKING**
- **OBSTACLE AVOIDANCE**
- **SEARCH AND PERSON RE-IDENTIFICATION**

Capítulo I

Introducción

Antecedentes

En la última década se ha producido un creciente interés en los vehículos aéreos no tripulados (UAV) en vista de las múltiples aplicaciones que estos ofrecen, cuyo bajo costo de adquisición y su amplio campo de investigación justifican su demanda hoy en día. Su empleo va desde ámbitos civiles hasta militares, entre ellos la agricultura, búsqueda y rescate de vidas, respuesta en situaciones de crisis, escaneo y cobertura de áreas determinadas, inspecciones de vigilancia, seguridad (Hyun, Apvrille, & Dugelay, 2014), (Garcia J. , 2016), (Andrea, Byron, Jorge, Inti, & Aguilar, 2018), (Pardo, Aguilar, & Toulkeridis, 2017), (Jara-Olmedo A. , Medina-Pazmiño, Tozer, Aguilar, & Pardo, 2018), (Jara-Olmedo A. , et al., 2018), (Orbea, et al., Vertical take off and landing with fixed rotor, 2017), (Amaguaña, Collaguazo, Tituaña, & Aguilar, 2018) realización de tareas tales como la identificación de grupos de personas entre otros. Cada una de estas misiones son llevadas a cabo en distintos entornos y es ahí donde surge el desafío de dotar a los UAVs con una mayor adaptabilidad y autonomía lo cual ha sido motivo de desarrollo de diferentes algoritmos, estrategias y técnicas de visión artificial para su control. No obstante, la navegación autónoma sigue siendo un desafío en fase de investigación.

Existen UAVs comerciales equipados con sensores que permiten parcialmente navegar en distintos entornos, sin embargo, siguen dependiendo de la experticia del operador. En este sentido uno de los problemas a resolver es realizar mediante la navegación autónoma el seguimiento de objetos en entornos complejos que pueden ocasionar colisiones, para ello el sistema debe estar preparado y ser capaz de evadir

obstáculos durante tal seguimiento. Estas características son fundamentales y claves en la navegación autónoma de UAVs.

Un segundo problema a resolver surge al momento de perder de vista al objetivo a seguir, de tal modo que al encontrarse en este estado el UAV deberá contar con la capacidad de buscarlo. Una vez cubierta esta problemática se le permitirá al UAV desenvolverse en múltiples entornos, reaccionar a distintas situaciones e identificar por sí mismos lugares u objetos con los cuales interactuar. El éxito del sistema planteado estará sujeto a la adquisición y procesamiento suficiente de datos.

En cuanto a la primera problemática algunos de los trabajos de investigación desarrollados en el área de visión computacional tanto para la navegación, detección y evasión de obstáculos, se basan en algoritmos con bibliotecas de visión artificial como OpenCV (Abdor & Delgado, 2017) y en implementaciones de algoritmos de extracción de características en imágenes, conocidos también como puntos de interés (Aguilar & Angulo, Real-time video stabilization without phantom movements for micro aerial vehicles, 2014), (Aguilar & Angulo, Real-Time Model-Based Video Stabilization for Microaerial Vehicles, 2016), (Aguilar & Angulo, Estabilización robusta de vídeo basada en diferencia de nivel de gris, 2013), (Aguilar & Angulo, Compensación y aprendizaje de efectos generados en la imagen durante el desplazamiento de un robot, 2012), (Aguilar & Angulo, Compensación de los efectos generados en la imagen por el control de navegación del robot Aibo ERS 7, 2012). Uno de ellos es SIFT, que entre sus múltiples usos está el estimar el tiempo de colisión en la tarea de evadir obstáculos (Badrloo, Samira, & Varshosaz, 2017). Otras alternativas son el uso de SURF (Aguilar, Casaliglla, Pólit, Abad, & Ruiz, 2017) y ORB cuya implementación dio la capacidad de distinguir los puntos de un obstáculo más cercano. Por otro lado, en (Alvarez, 2018) se usan técnicas

y algoritmos para la detección y estimación de proximidad de objetos basados en técnicas como Optical Flow.

En torno a la navegación autónoma y la detección de obstáculos se han obtenido buenos resultado aplicando técnicas como SLAM (localización y mapeo simultáneo) (Mur-Artal & Tards, 2017) para mapear un entorno desconocido, que simultáneamente puede llevar cámaras RGB-D (Aguilar, et al., Real-Time 3D Modeling with a RGB-D Camera and On-Board Processing, 2017), (Aguilar, et al., On-Board Visual SLAM on a UGV Using a RGB-D Camera, 2017), (Aguilar, et al., Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing, 2017), (Basantes, et al., 2018), (Aguilar, Cobeña, Rodriguez, Salcedo, & Collaguazo, 2018), cámaras para estéreo visión y un variedad de sensores ultrasónicos u otros elementos, demandando una alta capacidad de procesamiento y algunos costos extras. En (Gageik, Benz, & Montenegro, 2015) se propone el uso de escáneres láser LIDAR o conjuntos de sensores complejos mencionados en (Garcia, et al., 2016) que representan un peso extra para el UAV y generan un alto costo de adquisición. En favor de contrarrestar estos problemas se han presentado algoritmos de bajo costo computacional para el control adaptativo de la prevención de obstáculos en ambientes reales (Devos, Ebeid, & P, 2018).

En otro escenario se encuentra el seguimiento de objetivos, en donde algunos sistemas de navegación emplean las imágenes obtenidas desde una cámara abordo del UAV y con ello se han aplicado métodos de control Servo Visual Basado en Posición (PBVS), pero su ineficacia aparece en presencia de ruido de imagen. En (Corke, 2011) se presentan métodos de Servo Visual Basado en Imágenes (IBVS) que requiere menos cálculos y lo cual los hace más ágiles y robustos que el PBVS. Dentro de este esquema se encuentran también proyectos enfocados en el seguimiento de objetivos móviles

aplicando una red neuronal artificial con una función de base radial (RBF) y con un controlador adaptativo indirecto (Shirzadeh M., 2015), empleándola además en ambientes desconocidos (Shirzadeh, 2017).

Como ultima consideración esta la búsqueda de objetivos, encontrando así proyectos basados en la intercomunicación entre sistemas GPS y robots, apuntando a que dos robots autónomos jueguen a las escondidas (Hide and Seek) (Kleiven, Liu, McVeety, & Liu, 2012). Algunos trabajos de investigación han desarrollado estrategias para búsqueda y planeación de trayectorias de robots móviles, empleando campos potenciales artificiales basados en el posicionamiento y estimación futura de las trayectorias de los obstáculos móviles detectados, mediante un seguimiento con el uso filtros de Kalman (Mora & Tornero, 2007) o con enjambres de partículas activas brownianas (Cuchango, Helbert, Sofrony, & Jorge, 2012).

Dentro de los ámbitos tratados anteriormente, existen métodos basados en sistemas de bajo costo con UAVs, apoyados únicamente el uso de una cámara monocular, algunos de ellos se basan en un algoritmo de retroproyección de histograma (Valencia & Kim, 2018), dándole también un enfoque en la detección de personas mediante el histograma de gradientes (HOG) (Hyun, Apvrille, & Dugelay, 2014) agrupando así adultos y niños. Por otro lado existen métodos basados en Deep estimación o aprendizaje profundo para detectar obstáculos, los cuales requieren un previo entrenamiento que puede ser en base a redes neuronales (Shirzadeh M., 2015) (Shirzadeh, 2017) permitiendo a un micro-UAV reconocer, seguir y predecir la profundidad de ciertos objetos, tales como una persona, sillas, un balón de fútbol, etc., (Sumimoto, Miyata, Yoshimura, Uwate, & Nishio, 2018). Tal aprendizaje se lo ha hecho también con imágenes aéreas ocupando ya sea algoritmos de redes residuales

convolucionales (FCRN) (Laina, Rupprecht, Belagiannis, Tombari, & Navab, 2016) o algoritmos de estimación de profundidad monocular basados en CNN (Quisaguano, 2018) (Aguilar, Quisaguano, Alvarez, Pardo, & Proaño, 2018). Otra alternativa para la detección de objetos es el uso de herramientas como YOLO (You Only Look Once) (Redmon & Farhadi, 2018) que sigue mostrando resultados muy prometedores.

Estas técnicas han sido utilizadas en diferentes tipos de micro-UAVs, algunos de ellos son los drones Parrot, que ofrecen muchas facilidades al emplear un sistema operativo de código abierto ROS (Robot Operating System) y permiten integrar aplicaciones para estructurar sistemas complejos (Quisaguano, 2018) (Albornoz & Calahorrano, 2016). El mismo será utilizado para desarrollar el presente trabajo de investigación haciendo frente a las problemáticas planteadas.

Justificación e Importancia

En los últimos años múltiples aplicaciones en el campo de UAVs se han centrado en el uso de sensores de visión e inteligencia artificial, la cual puede ser una alternativa para percepción, navegación y localización, dado que operarlos no es tarea sencilla para cualquier usuario y en ocasiones es mandatorio contar con un piloto calificado.

Con el mercado de drones creciendo rápidamente, crece aún más el desafío de obtener sistemas con mayor autonomía y capacidad de adaptación a distintos entornos. Gran parte de los métodos convencionales se han enfocado en la combinación y equipamiento de múltiples sensores, ocasionando peso extra en el UAV y viéndose así condicionados por la limitante carga útil que puede transportar, tal es el caso de sistemas sónicos, edometría estéreo y Sistemas laser mini LIDAR (Garcia J. , 2016) que a su vez pueden llegar a representar un alto costo en su adquisición.

Para hacer frente a este problema se encuentra ya desarrollada una tesis la cual dota de cierta autonomía a un micro-UAV a través de algoritmos de estimación de profundidad en imágenes proporcionadas por la propia cámara incorporada en el vehículo. Su enfoque es la evasión de obstáculos y seguimiento de un objetivo (Quisaguano, 2018) eliminando así el costo en la adquisición de sensores u otros implementos que puedan adicionar peso extra e incidir sobre su consumo normal de batería. Tal sistema requiere de un entrenamiento previo y demoroso con las imágenes del objetivo a seguir, además no cuenta con un algoritmo que actúe al momento de perder de vista al objetivo, de modo que en este estado el sistema se queda paralizado. No obstante, en la presente tesis a desarrollarse se pretende usar una diferente estrategia en cuanto a la detección de objetos y seguimiento de una persona, tratando adicionalmente de dotar al sistema de la capacidad de búsqueda en el caso de perderla de vista, para ello se hará uso de diferentes algoritmos juntamente con el controlador más adecuado ya sea PI, PD o PID, cuya elección dependerá de la caracterización de la planta.

Los aplicativos en esta temática van desde ámbitos civiles hasta militares, entre ellos se encuentra la Universidad de las Fuerzas Armadas, que por medio de los centros de investigación CICTE y CIDFAE iniciaron el desarrollo del Proyecto de “Sistemas inteligentes de monitoreo y aterrizaje para UAVs tácticos y mutirotores Fase 1 - SmartDrone1”. En él se busca incrementar la eficiencia de las diversas operaciones militares que realizan, tales como tareas de búsqueda, rescate, operaciones contraterrorismo y operaciones regulares de defensa apoyados en el uso de UAVs.

Alcance del Proyecto

El presente proyecto de investigación tiene como objetivo el desarrollo de un sistema autónomo con un micro-UAV basado en algoritmos de detección y estimación de profundidad monocular para la evasión de obstáculos, además de algoritmos de seguimiento y búsqueda de un objetivo dependiendo solamente del uso de la cámara incorporada en el mismo, cuyo procesamiento se lo realizará desde una estación en tierra.

La primera fase del proyecto se centrará en la etapa de percepción, esta constará del estudio de técnicas a través de Tensorflow y YOLO para determinar la estimación de profundidad monocular y reconocimiento de objetos en entornos complejos.

La siguiente fase dará lugar a la integración de ambas técnicas ya mencionadas a fin de desarrollar la etapa del control de navegación. Esta constará del diseño de un controlador PID para el seguimiento de objetivos.

Posteriormente se hará una investigación sobre algoritmos de Re-Identificación enfocados a la etapa de búsqueda de una persona. Con ello se llevará acabo la planeación y búsqueda del objetivo al perderlo de vista, conjuntamente con el seguimiento y evasión de obstáculos, tomando como referencia el juego Hide and Seek (Las escondidas). Todo este sistema será implementado en el entorno ROS juntamente con el driver de control para “Parrot Bebop drones” (cuadricópteros).

En la etapa final del proyecto se dará lugar a la realización de pruebas de funcionamiento que busca analizar la precisión en la detección y la velocidad de respuesta del sistema, tales pruebas tendrán lugar en entornos complejos.

Objetivos

Objetivo General

Desarrollar un sistema en el framework ROS para el seguimiento y búsqueda de objetivos en entornos complejos usando micro vehículos aéreos y técnicas de estimación de profundidad en imágenes monoculares, para aplicaciones militares tales como el apoyo en la toma de decisiones en tiempo real en operaciones urbanas

Objetivos Específicos

- Realizar un estudio del estado del arte sobre algoritmos de estimación de profundidad en imágenes monoculares y algoritmos enfocados en la detección, evasión de obstáculos, seguimiento y búsqueda de objetivos.
- Desarrollar el sistema de detección y reconocimiento de objetivos mediante algoritmos basados en visión por computador.
- Desarrollar el sistema de evasión de obstáculos mediante estimación de profundidad en imágenes monoculares, además del seguimiento y búsqueda de objetivos.
- Desarrollar el algoritmo para el sistema de control de vuelo autónomo de un micro-UAV integrando la detección, el seguimiento, la evasión de obstáculos, y la búsqueda de un objetivo.
- Realizar la evaluación del desempeño del sistema en diferentes entornos.

Capítulo II

Fundamentación Teórica

Vehículos aéreos no tripulados

Denominado *UAVs* por sus siglas en inglés *(Unmanned Aerial Vehicle)*. En estos vehículos el vuelo puede ser operado remotamente por un humano o de forma autónoma, ya sea que lleve una serie de procesadores abordo o que todo este proceso se lo haga desde una estación en tierra. Con ello surge el término conocido como *UAS*, que hace referencia a un sistema que comprende un *UAV*, una estación de control en tierra y un sistema de comunicación entre ellos (Colomina & Molina, 2014).

Al ser plataformas de bajo costo facilitan la investigación y el desarrollo de aplicativos en diferentes áreas, tanto en el ámbito militar como en el civil, entre ellas están sistemas para detección de objetos y evasión de obstáculos (Quisaguano, 2018) (Calderón, Aguilar, & Merizalde, 2020) (Alvarez, 2018) (Aguilar, et al., Pedestrian Detection for UAVs Using Cascade Classifiers and Saliency Maps, 2017),(Aguilar, et al., Real-Time Detection and Simulation of Abnormal Crowd Behavior, 2017),(Aguilar & Angulo, Estabilización de vídeo en micro vehículos aéreos y su aplicación en la detección de caras, 2014),(Aguilar, et al., 2017),(Galindo, Aguilar, & Reyes Ch, 2019) (Badrloo, Samira, & Varshosaz, 2017), seguimiento de objetos (Abdor & Delgado, 2017) (Albornoz & Calahorrano, 2016) lo cual deriva a aplicativos de seguridad, vigilancia y detección de riesgos como los mencionados en (Morales & Paucar, 2017) (Hyun, Apvrille, & Dugelay, 2014).

En base a como están clasificados los *UAVs* (Quisaguano, 2018), está establecido para este proyecto de investigación el uso de un vehículo aéreo de micro escala conocido

también como *MAV´s* (Grijalva, 2019), especialmente el enfoque será a un *micro-UAV* con una configuración de cuatro rotores o cuadricóptero.

Dinámica del movimiento de un cuadricóptero

El movimiento de un cuadricóptero se basa en la variación angular de cada uno de los cuatro motores para realizar desplazamientos lineales y angulares. Esta variación está muy ligada a los ángulos de navegación *pitch*, *roll* y *yaw* que describen la orientación del *UAV* y que permiten que el cuadricóptero pueda ser maniobrado y programado para realizar diferentes comportamientos autónomos (Aguilar, Salcedo, Sandoval, & Cobeña, 2017),(Grijalva & Aguilar, 2019),(Orbea, et al., Math Model of UAV Multi Rotor Prototype with Fixed Wing Aerodynamic Structure for a Flight Simulator, 2017), (Aguilar W. G., Angulo, Costa, & Molina, 2014),(Aguilar, Angulo, & Pardo, Motion intention optimization for multirotor robust video stabilization, 2017).

Un punto importante en la dinámica es establecer la distribución de los ejes en el sistema de referencia móvil, tal sistema irá acorde a la configuración del cuadricóptero, puede ser esta en cruz (x) o positiva (+). En la Figura 1 se presenta la configuración en cruz usada para la ejecución de este proyecto de investigación. En ella también se ilustra la influencia de la velocidad de giro de cada motor en la dinámica del movimiento, al igual que la fuerza de empuje producida en cada uno de ellos, siendo esta proporcional al cuadrado de la velocidad angular:

$$f_i = kw_i^2 \tag{1}$$

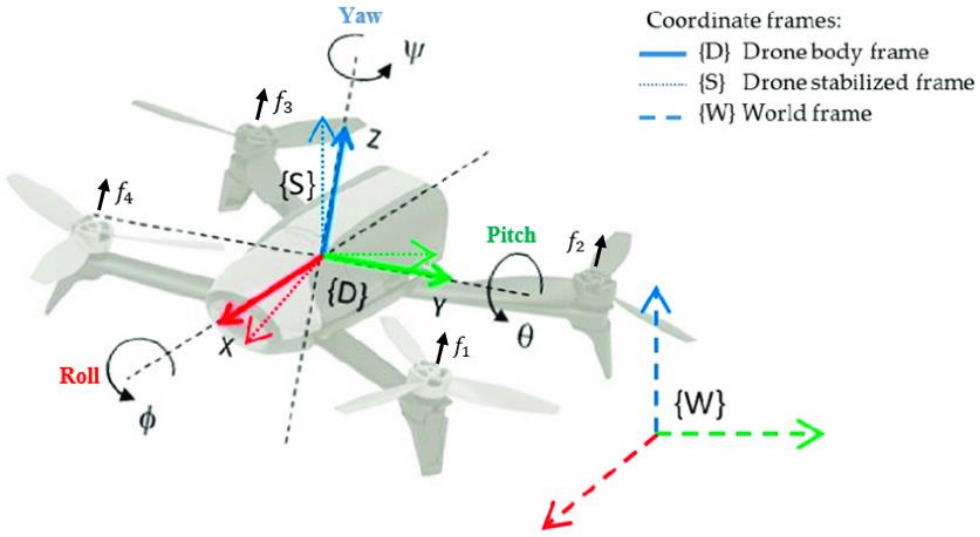
$$\tau = \left(\tau_{pitch} \ \tau_{roll} \ \tau_{yaw}\right)^2 \tag{2}$$

La diferencia de estas fuerzas de empuje genera tres momentos para cada par de torsión relacionado con los ángulos de navegación.

Cada hélice está configurada de forma alternada para realizar su giro hacia un solo sentido, es decir la hélice 1 y su opuesta la hélice 3 giran en sentido horario, por otro lado, las hélices 2 y su opuesta la hélice 4 giran en sentido antihorario.

Figura 1

Sistema de referencia móvil para una configuración en cruz



Nota. El gráfico representa la configuración de los ejes y ángulos de navegación del micro-UAV. Tomado de *A Multi-Sensorial Simultaneous Localization and Mapping (SLAM) System for Low-Cost Micro Aerial Vehicles in GPS-Denied Environments,* (López, et al., 2017).

- **Pitch:** Rotación alrededor del eje y, la inclinación en este eje permitirá que el cuadricóptero se desplace hacia el frente o hacia atrás.
- **Roll:** Rotación alrededor del eje x, la inclinación en este eje permitirá que el cuadricóptero se desplace lateralmente.

- **Yaw:** Rotación alrededor del eje z, la rotación en este eje permitirá que el cuadricóptero gire sobre su propio eje.

Los movimientos de rotación y desplazamiento se generan a partir de diferentes secuencias (Salcedo, 2018) presentadas en la Tabla 1, referentes a una configuración en cruz (x) y a la numeración de los rotores de la Figura 1.

Tabla 1

Secuencias para la generación de movimientos en la configuración cruz (x)

Rotación	Movimiento	Rotor 1	Rotor 2	Rotor 3	Rotor 4
Pitch	Hacia adelante	$w_1 - \Delta$	$w_2 + \Delta$	$w_3 + \Delta$	$w_4 - \Delta$
	Hacia atrás	$w_1 + \Delta$	$w_2 - \Delta$	$w_3 - \Delta$	$w_4 + \Delta$
Roll	Hacia la derecha	$w_1 + \Delta$	$w_2 + \Delta$	$w_3 - \Delta$	$w_4 - \Delta$
	Hacia la izquierda	$w_1 - \Delta$	$w_2 - \Delta$	$w_3 + \Delta$	$w_4 + \Delta$
Yaw	Giro horario	$w_1 - \Delta$	$w_2 + \Delta$	$w_3 - \Delta$	$w_4 + \Delta$
	Giro antihorario	$w_1 + \Delta$	$w_2 - \Delta$	$w_3 + \Delta$	$w_4 - \Delta$
Ninguna	Hacia arriba	$w_1 + \Delta$	$w_2 + \Delta$	$w_3 + \Delta$	$w_4 + \Delta$
Ninguna	Hacia abajo	$w_1 - \Delta$	$w_2 - \Delta$	$w_3 - \Delta$	$w_4 - \Delta$

Nota. Esta tabla muestra las secuencias que genera el micro-UAV en sus rotores para realizar los distintos movimientos.

Control de vuelo de un Vehículo aéreo no tripulado

El sistema de control de vuelo de un *UAV* es una tarea desafiante debido a la dinámica no lineal y la configuración en su diseño, teniendo que realizar acciones de control tales como el ascenso y descenso del cuadricóptero, el cabeceo (rotación en pitch), balanceo (rotación en roll) y la guiñada (rotación en yaw).

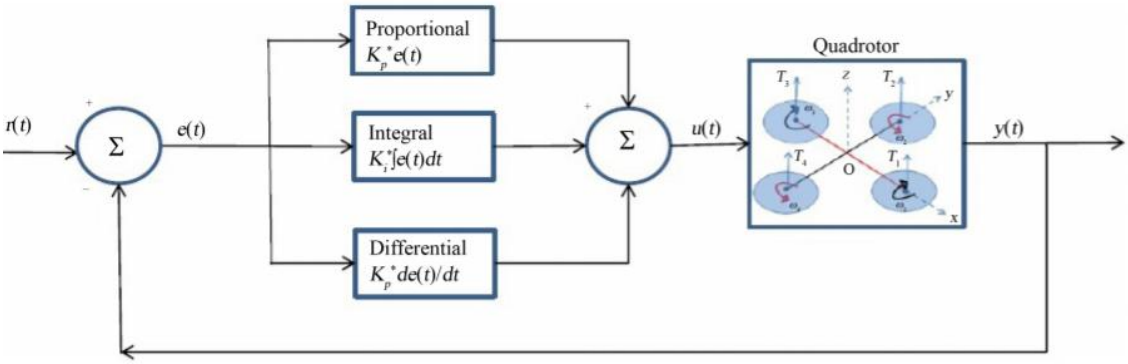
Si bien es cierto existen una variedad de controladores, pero el control lineal clásico *PID* sigue predominando (Aguilar, Casaliglla, & Pólit, Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles, 2017),(Aguilar, Álvarez, Grijalva, & Rojas,

2019),(Aguilar, et al., 2018), (Aguilar, Manosalvas, Guillén, & Collaguazo, 2018),(Aguilar, Angulo, & Costa-Castello, Autonomous Navigation Control for Quadrotors in Trajectories Tracking, 2017), de tal forma que es usado en una amplia gama de aplicaciones, siendo así el más implementado y preferido en la industria, por su robustez, diseño simple, precisión y facilidad de ajuste de sus parámetros que a menudo se sintonizan experimentalmente.

El esquema de la Figura 2 representa un controlador *PID*, mismo que ha sido aplicado con éxito en cuadricópteros, teniendo buenos desempeños en el seguimiento del ángulo de cabeceo (Zulu & John, 2014).

Figura 2

Diagrama de bloque de un controlador pid para un cuadricóptero



Nota. El gráfico representa la estructura general de un sistema de control PID para ser implementado en un micro-UAV.

Formación y representación de una imagen

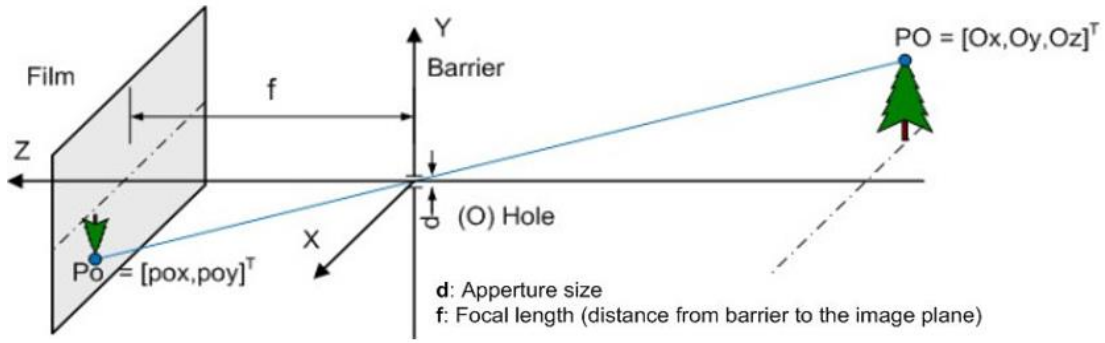
Una de las formas en las que un *UAV* puede percibir su entorno es mediante el uso de sensores pasivos (cámaras monoculares) como un dispositivo de visión, captando los rayos de luz reflejados en los objetos y pretendiendo así simular la visión humana.

A través de un modelo frecuentemente usado en visión por computador llamado pinole mostrado en la Figura 3, se puede obtener una aproximación matemática a la representación de una escena en tres dimensiones del mundo real a una imagen en dos dimensiones.

Este modelo se fundamenta en el principio de proyección en perspectiva, en vista de que los rayos de luz pasan por medio de un único punto en la cámara, conocido como centro óptico, que los proyecta a un plano de la imagen. Cada rayo de luz proyectado incide sobre una celda o elemento fotodetector de una matriz CCD (*Charge-coupled device*)

Figura 3

Modelo de cámara pinhole



Nota. El gráfico muestra el proceso de captación de una imagen en una cámara. Tomado de *Camera Models and Fundamental Concepts Used in Geometric Computer Vision,* (Sturm, Ramalingam, Tardif, Gasparini, & Barreto, 2011).

En este modelo hay una relación lineal entre la dirección del rayo y la posición del punto de la imagen, que se expresa en una matriz de parámetros intrínsecos o matriz de calibración (Garcia J. , 2016). Donde el parámetro f es la distancia focal medida en

dimensiones de píxeles y las coordenadas (x_0, y_0) representan el punto principal de la proyección en el plano de la imagen.

$$K = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Para formar las imágenes rápidamente es necesario el uso de lentes en las cámaras para tener una mayor concentración de la luz incidente. No obstante, el uso de estos elementos introduce distorsiones, entre las más importantes están las distorsiones radiales y tangenciales para las cuales ya se han generado varias soluciones para contrarrestar su efecto.

Inteligencia Artificial

La *IA* es la forma en como una maquina imita comportamientos o realiza tareas aparentemente inteligentes, es decir una tarea que bien puede ser aprendida o preprogramada.

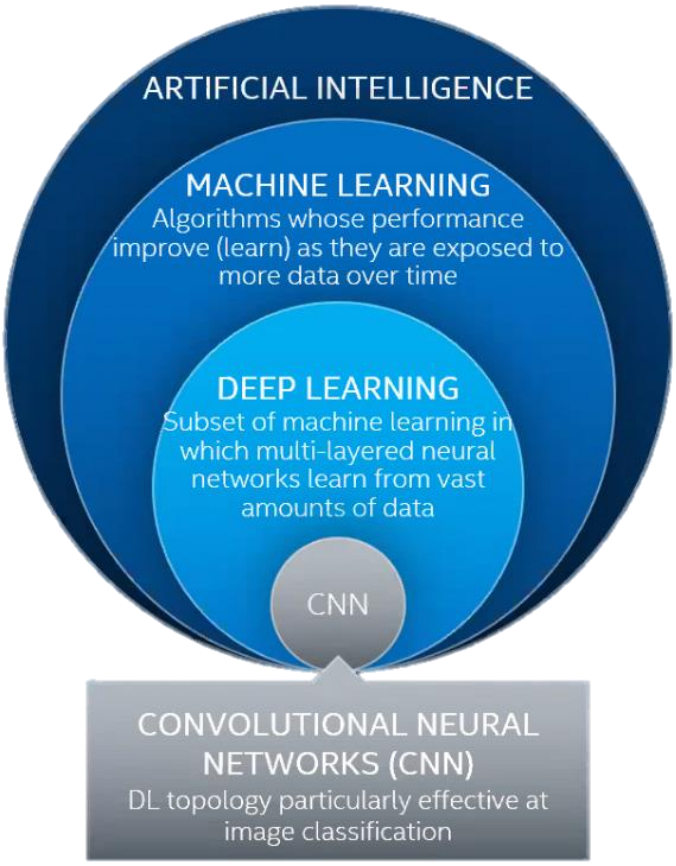
Los grandes avances en esta tecnología abarcan una serie de campos que apuntan a varias habilidades asociadas con la inteligencia, algunas de ellas son:

- Computer Vision: Cuando la tarea que resuelve de forma aparentemente inteligente es la de observar y entender el mundo por medios visuales. Habilidad de ver.
- Robótica: Si la tarea tiene que ver con dar la habilidad de movilidad a un robot.
- NLP (*Natural Language Processing*): Cuando se requiere interpretar, entender y procesar el Lenguaje.
- Speech: Cuando la tarea es el reconocer el habla, poder decodificar esos sonidos en palabras.

Un elemento identificado con la inteligencia que habilita un nuevo campo en esta área y define a un agente como inteligente es la capacidad de aprender *(Machine Learning).* Esta es la forma más común de lograr inteligencia artificial hoy en día (Mark Robins, 2020). A través de la Figura 4 se puede comprender la relación existente entre IA, el aprendizaje automático *(Machine Learning)* y el aprendizaje profundo *(Deep Learning)* como un tipo especial de aprendizaje automático.

Figura 4

Relación entre inteligencia artificial, machine learning y deep learning



Nota. El gráfico representa los tipos de aprendizaje que engloba la inteligencia artificial. Tomado de *The Difference Between Artificial Intelligence, Machine Learning and Deep Learning*, (Mark Robins, 2020).

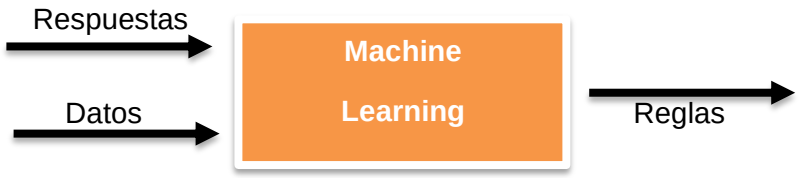
Machine Learning

El aprendizaje automático es considerado como una tecnología altamente revolucionaria, siendo un componente fundamental dentro del campo de la inteligencia artificial, transversal a todos los demás, en el que una maquina con la capacidad de entender información de entrada y procesarla, cuenta además con la habilidad de aprender esa tarea evolutivamente o por medio de ensayo y error. La clave es la generalización del conocimiento a partir de un conjunto de experiencias.

Tradicionalmente la programación ha consistido en generar reglas para que estas actúen sobre los datos y a partir de ello generar una respuesta. Pero el machine learning le da un cambio a este esquema mostrado en la Figura 5 y en vez de que el programador determine las reglas, ahora esa tarea pasa a ser del ordenador, que con tan solo darle como entrada los datos y las respuestas deseadas, la computadora infiere cuales son estas reglas o patrones (TensorFlow.org, 2019).

Figura 5

Esquema General del Machine Learning



Nota. El gráfico muestra el proceso de un sistema con machine learning.

Las reglas obtenidas como resultado, son consideras como un modelo pre-entrenado, el esquema de ello se ilustra en la Figura 6, y es aquí donde utilizará estas reglas para obtener predicciones con nuevos datos dados.

Figura 6

Esquema general de un modelo pre-entrenado

Nota. El gráfico muestra el proceso de un modelo pre-entrenado con machine learning.

Paradigmas de Aprendizaje

- **Supervisado:** Aprendizaje que se encarga en descubrir las relaciones o patrones entre una variable de entrada y una salida. Esto surge de enseñarle a estos algoritmos el resultado a obtener para un determinado valor.
- **No supervisado:** Por otro lado, este aprendizaje consigue adquirir conocimiento únicamente de los datos proporcionados como entrada, sin necesidad de que en algún momento se le indique al sistema el resultado que se desea obtener. Dentro de este campo se encuentra el problema de clusterización, donde el algoritmo encuentra patrones de similitud entre los datos.
- **Reforzado:** Aprendizaje orientado a objetivos, en el que los agentes aprenden a resolver diferentes tareas en función de los estado actuales y anteriores, a fin de tomar las decisiones más adecuadas para maximizar la recompensa. La única supervisión dada por el sistema en este tipo de aprendizaje es una señal de castigo o recompensa

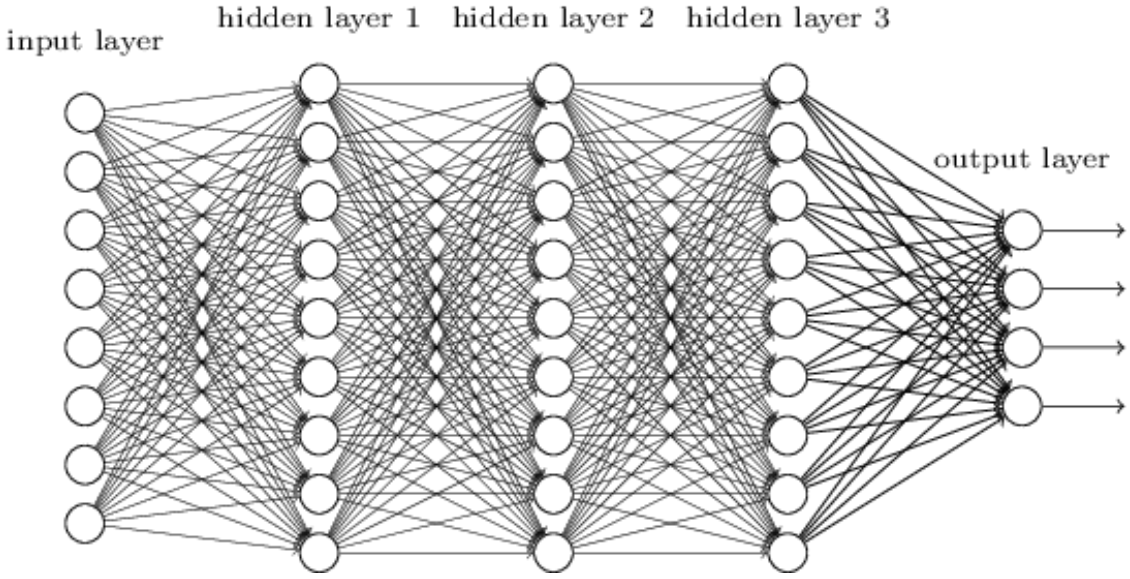
Redes Neuronales

Están representadas por una interconexión de neuronas a través de una serie de capas ocultas, donde cada neurona internamente cuenta con una función de activación encargada de añadir a los valores de salida deformaciones no lineales. Esta unión de

neuronas forma capas neurales totalmente conectadas tal como se muestra en la Figura 7, donde el valor de salida en cada capa alimenta a las siguientes, hasta llegar a la capa output, que muestra las predicciones finales con los puntajes de clase para una entrada dada (Nielsen, 2019). Cada una de las neuronas dentro de una capa actúan de manera independiente y no comparten ninguna información.

Figura 7

Red neuronal totalmente conectada



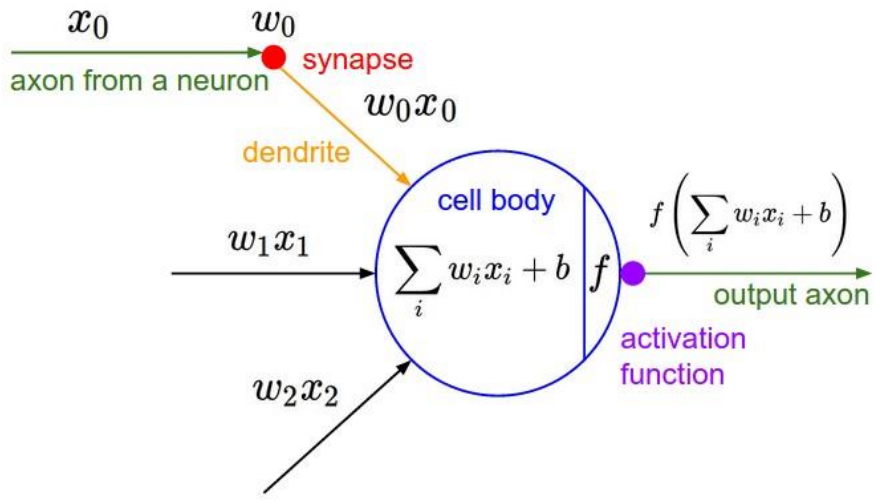
Nota. El gráfico muestra la anatomía de una red totalmente conectada. Tomado de *Neural Networks and Deep Learning*, (Nielsen, 2019).

Son muy usadas en tareas de reconocimiento y clasificación de: imágenes, videos, texto, audio etc. Las limitaciones de estas redes es el no adaptarse a imágenes completas en vista de que demandan un alto costo computacional por a la excesiva cantidad de parámetros que debe manejar, lo que puede conducir rápidamente aun sobreajuste.

- **Neurona**: Conocida desde los años 50 también como *Perceptrón*, mostrado en la Figura 8. Esta recibe estímulos externos a través de conexiones de entrada con sus respectivos pesos e internamente una función similar a un modelo de regresión lineal (para arquitecturas de una única neurona, problemas lineales) o una función de activación (al trabajar con capas de neuronas para abarcar problemas no lineales) es quien se encarga de generar un valor de salida a partir de las entradas ponderadas.

Figura 8

Estructura de un Perceptrón

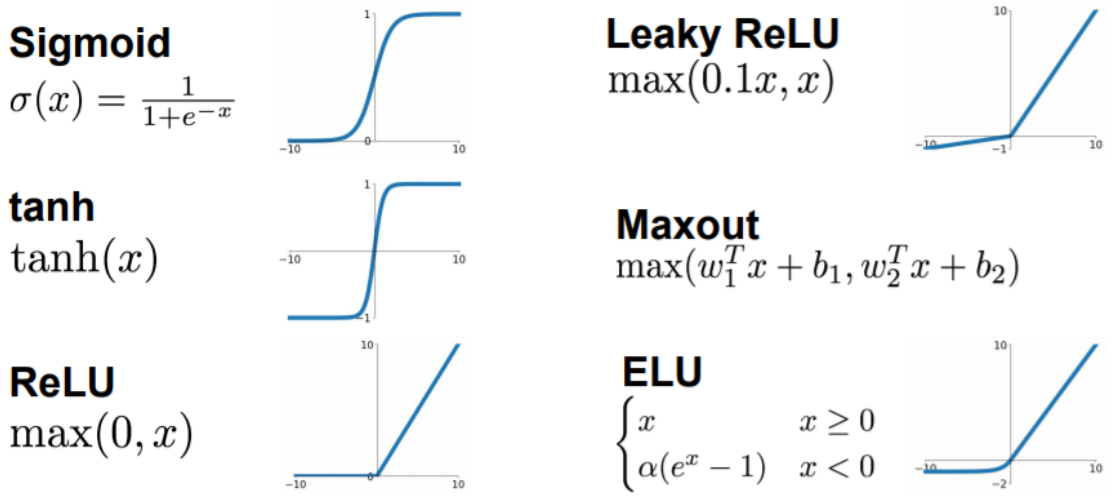


Nota. El gráfico muestra a detalle los componentes de una neurona artificial. Tomado de *CS231n: Convolutional Neural Networks for Visual Recognition*, (Stanford, 2020).

- **Funciones de Activación**: Estas funciones matemáticas son las que determinan los valores de salida de la red neuronal, estas ayudan a normalizar estos valores, activando o desactivando la salida en función de un umbral. Pueden ser lineales o no lineales, las más conocidas se muestran en la Figura 9.

Figura 9

Funciones de Activación.



Nota. El gráfico muestra algunas de las funciones de activación usadas al interior de una neurona para evaluar la suma ponderada de las entradas. Tomado de *CS231n: Convolutional Neural Networks for Visual Recognition*, (Stanford, 2020).

- **Backpropagation**: Es la forma en como las redes neuronales ajustan sus parámetros para así aprender automáticamente la información que está procesando. Esta técnica se basa en la retropropagacion de errores desde la salida hacia las capas anteriores, esto lo hace usando el descenso del gradiente para optimizar la función de coste y juntamente con la técnica de backpropagation se encuentra el vector de gradientes el cual permite minimizar el error y entrenar a la red.

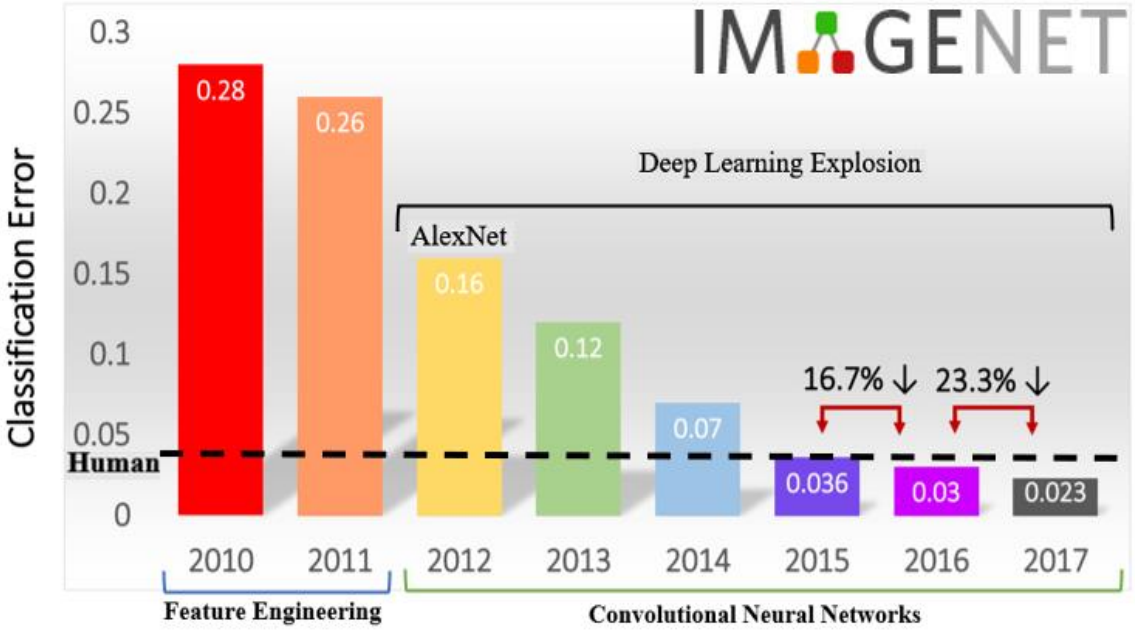
Deep Learning

Esta revolución del *Deep learning* surge el 2012 dando un salto cualitativo con la arquitectura *AlexNet* (Krizhevsky, Sutskever, & Hinton, 2012) que presentaba una solución basada en *redes neuronales convolucionales*, el cual obtuvo una ratio de error

por debajo del 25% en el *ImageNet* Challenge como se observa en la Figura 10, para ello usaron el dataset abierto de ImageNet que contiene más de 14 millones de imágenes.

Figura 10

Classification Challenge



Nota. El gráfico muestra los resultados del concurso ILSVRC 2017 de clasificación de imágenes entre los años 2010 y 2017. Tomado de *ImageNet Large Scale Visual Recognition Challenge,* (Russakovsky, et al., 2015).

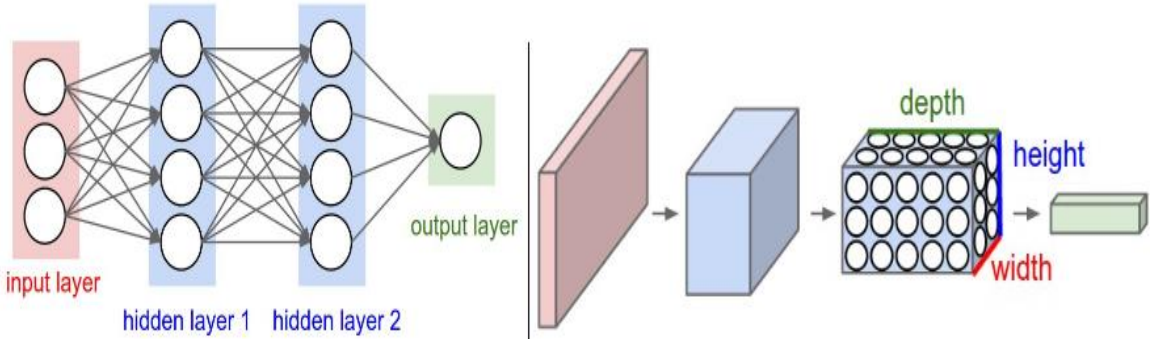
Redes Neuronales Convolucionales

Son también conocidas como *ConvNets*. Aprovechan que su input son imágenes y usan una estructura tridimensional con tres capas de redes neurales, cada dimensión se especializa en analizar uno de los canales RGB de la imagen como se muestra en la Figura 11 y en función de ello se extraen patrones y características importantes. Esta arquitectura fue propuesta en (Krizhevsky, Sutskever, & Hinton, 2012) llamada como *AlexNet*, similar a la red neuronal artificial de múltiples capas LeNet-5 (LeCun, Bottou,

Bengio, & Haffner, 1998). Tuvo una gran acogida gracias a que contiene una estructura más profunda, lo que mostró significativas mejoras en comparación a métodos anteriores en la tarea de clasificación de imágenes. Por su éxito son muy usadas en el campo de *computer vision* para procesar imágenes y videos en tareas de detección, identificación y clasificación.

Figura 11

Red Neuronal Fully Connected y Convolutional Neural Network

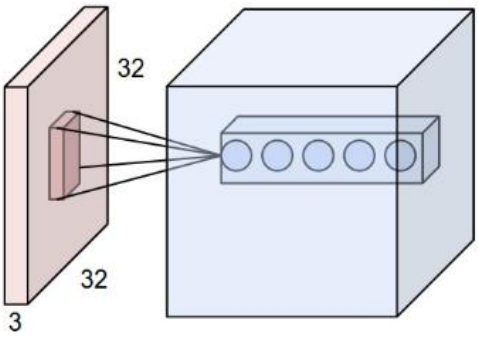


Nota. El gráfico muestra la diferencia entre Red Neuronal Fully Connected (Izquierda) y una Convolutional Neural Network/ConvNets (Derecha). Tomado de *CS231n: Convolutional Neural Networks for Visual Recognition*, (Stanford, 2020).

Dentro de una *red neuronal convolucional* las neuronas en una capa no tienen conexiones a todas las demás neuronas de la capa siguiente, que es lo que la diferencia de una red neuronal completamente conectada. En este tipo de redes como se muestra en la Figura 12, cada grupo de neuronas se enfocan en analizar específicamente una determinada región de la imagen, por ejemplo, cada grupo de neuronas se encargará de buscar ya sea rasgos de cabezas, extremidades, ojos, bocas, siluetas y demás.

Figura 12

Entrada analizada por un volumen determinado de neuronas



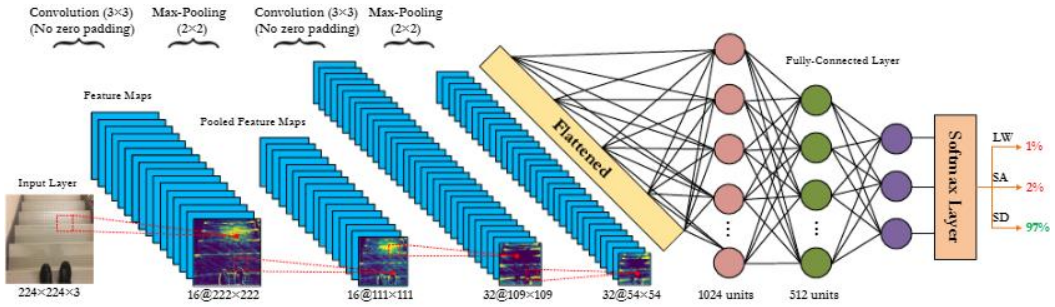
Nota. El gráfico muestra el proceso de análisis de una imagen, generando un volumen de neuronas con las características de esta. Tomado de *CS231n: Convolutional Neural Networks for Visual Recognition*, (Stanford, 2020).

Al final de toda la red, la salida es un vector de probabilidades de cada una de estas características que indican que tan probable es que cada una de ellas pertenezca a una determinada clase.

En un modelo bien entrenado la red neuronal debe caracterizarse por tener un sesgo bajo (el modelo genera predicciones correctas con los ejemplos entrenados) y una varianza baja (el modelo es capaz de generalizar su conocimiento). En la Figura 13 se muestra la arquitectura de una red neuronal convolucional:

Figura 13

Estructura de una Convolutional Neural Network/ConvNets

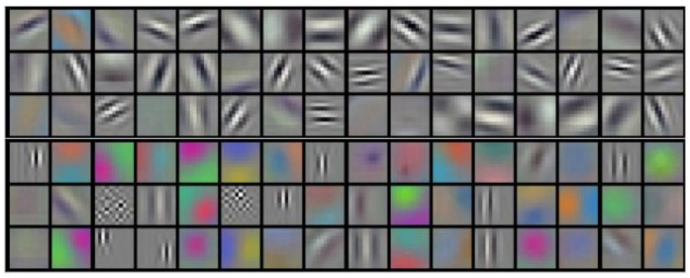


Nota. El gráfico muestra la estructura interna de una red neuronal convolucional. Tomado de *Convolutional Neural Network for Environmentally Aware Locomotion Mode Recognition of Lower-Limb Amputees,* (Khademi, Gholamreza; Simon, Dan, 2019).

- **Capas convolucionales:** Escanean la imagen mediante una ventada deslizante y se enfocan en el análisis de una pequeña parte de ella para así generar un mapa de características. Esto lo logra aplicando una serie de filtros a la imagen que extraen y resaltan distintas características y patrones. En la Figura 14 se observan varios ejemplos de estos filtros.

Figura 14

Ejemplos de Filtros aplicados a imágenes.

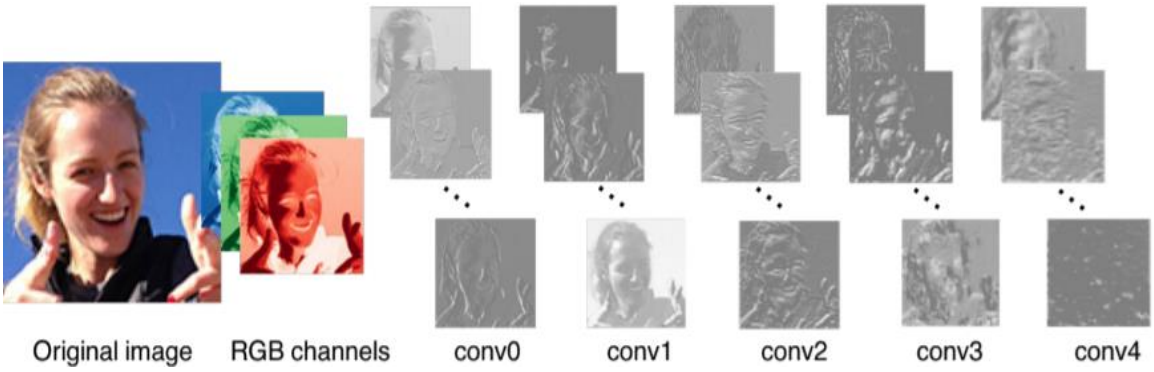


Nota. El gráfico muestra algunos de los filtros usados por las capas convolucionales. Tomado de *CS231n: Convolutional Neural Networks for Visual Recognition*, (Stanford, 2020).

Al aplicar esta serie de filtros sobre una imagen de entrada, se pueden obtener resultados como los que se observan en la Figura 15.

Figura 15

Resultado de aplicar Convoluciones a una imagen

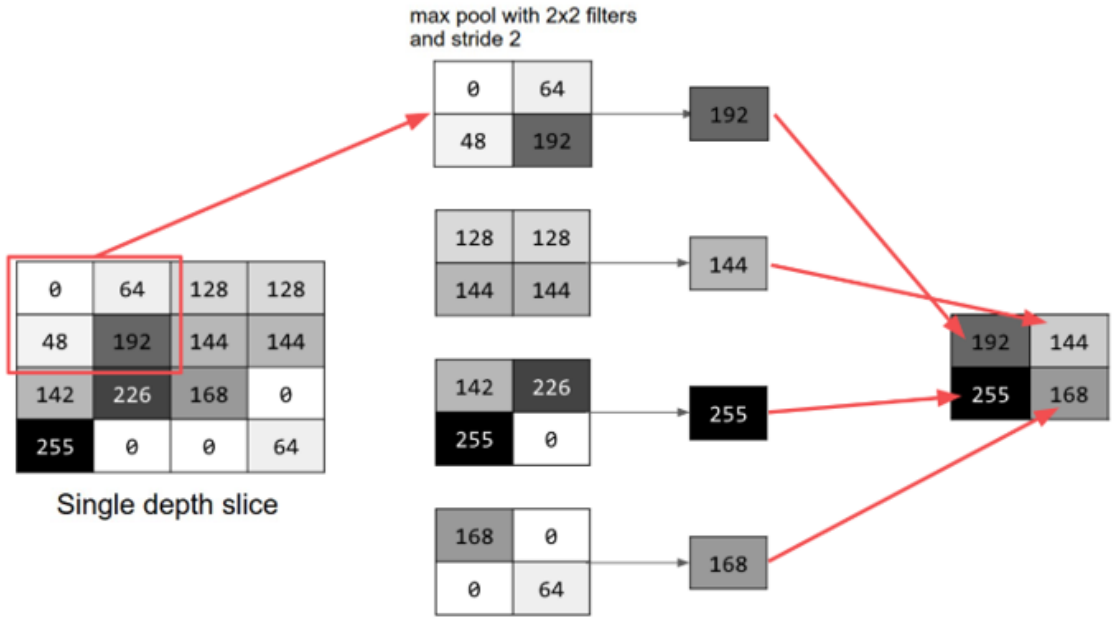


Nota. El gráfico muestra los resultados de las capas convolucionales al aplicar distintos filtros a la imagen. Tomado de *DeepFace: Closing the Gap to Human-Level Performance in Face Verification,* (Taigman, Yang, Ranzato, & Wolf, 2014).

- **Capas MaxPooling:** Estas capas son las encargadas de hacer un re-escalado de la imagen filtrada en las capas anteriores y hacen una compresión de ella, manteniendo el valor más alto de cada pixel, es decir la información de las características más importantes. De esta manera se reduce la cantidad de parámetros y permite controlar el sobreajuste.

Figura 16

Diagrama del proceso realizado por las capas Max Pooling



Nota. El gráfico muestra el proceso de re-escalado realizado por las capas max pooling. Tomado de *Learn Tensorflow 4: Convolutional Neural Networks (CNNs),* (TensorFlow.org, 2019).

- **Capa Flattened:** Transforma las imágenes expresadas en matrices bidimensionales de características provenientes de la capa anterior a un vector que alimenta a la siguiente capa, la capa de *redes neuronales fully connected*.
- **Capas Fully Connected:** Conocida también como capa densa, que cuenta con una colección de neuronas representando diferentes partes del objeto. En esta capa se recibe los resultados aplanados (vectores) de las capas convolucionales y genera a partir de ello una predicción.

- **Dropout:** Esta característica es usada durante el entrenamiento de una red y su función es desconectar ciertas neuronas en cada iteración para evitar el sobreajuste (*overfitting*).
- **Softmax:** Típicamente al final de la capa fully connected se usa una función de activación llamada Softmax, que es la responsable de manejar y clasificar los resultados en múltiples clases, normalizando la salida entre 0 y 1. También es común encontrar algoritmos *SVM* (Support Vector Machine) para realizar esta misma tarea.

El estado del arte en el área del deep learning es muy extenso y abarca diferentes tareas en cuanto al campo de *computer visión* se refiere, entre ellas: *Semantic Segmentation, Image Classification, Object Detection, Image Generation, Domain Adaptation, Pose Estimation, Denoising, Facial Recognition, Depth Estimation, Data Augmentation, Gesture Recognition* entre otros.

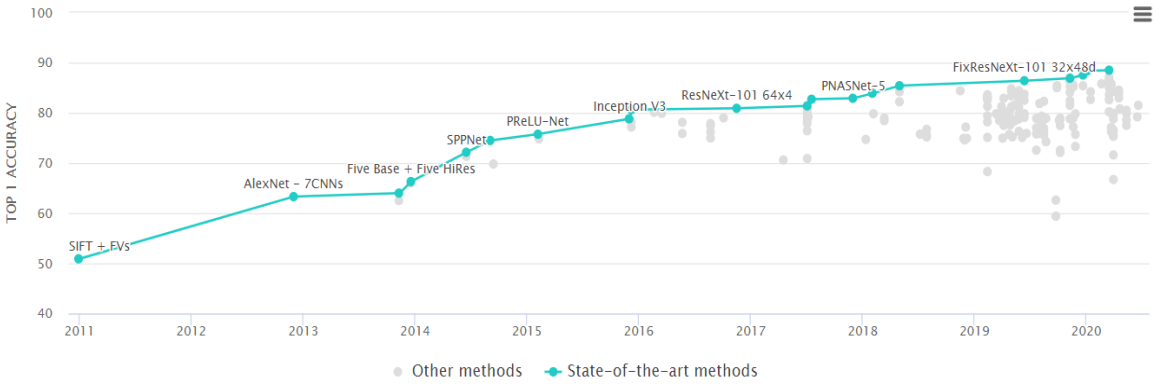
Arquitecturas de Redes Neuronales Convolucionales

A continuación, se mencionan los métodos y arquitecturas usados generalmente en tres de las muchas tareas que abarca el *deep learning*.

- **Clasificación de imágenes:** Los recientes avances en las redes neuronales convolucionales han generado una variedad de arquitecturas de *CNNs* (Gu, et al., 2018) enfocadas a esta tarea. Entre las más conocidas están *LeNet, AlexNet, ZFNet, VGGNet, GoogLeNet, ResNet, MobileNet, Darknet53, Inception*. En la Figura 17 se muestra el ranking de estas arquitecturas que son evaluadas en el dataset de *ImageNet*.

Figura 17

ImageNet - Ranking de arquitecturas para clasificación de imágenes

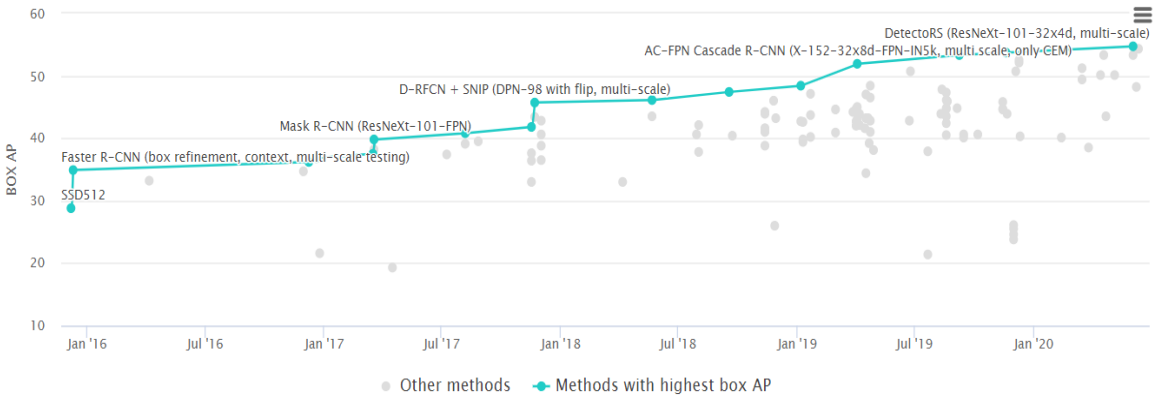


Nota. El gráfico muestra el ranking de modelos entrenados sobre el dataset ImageNet. Tomado de *Object Detection on COCO test-dev,* (paperswithcode, 2020).

- **Detección de objetos*:*** Generalmente hacen uso de las arquitecturas anteriores en combinación con otros métodos, entre las más conocidas *R-CNN, Fast R-CNN, Faster-RCNN, Mask R-CNN, YOLO, RetinaNet, EfficientNet, DetectoRS*, etc. Su evaluación y ranking es realizado sobre el dataset de *COCO*.

Figura 18

Coco test dev - Ranking de arquitecturas para detección de objetos

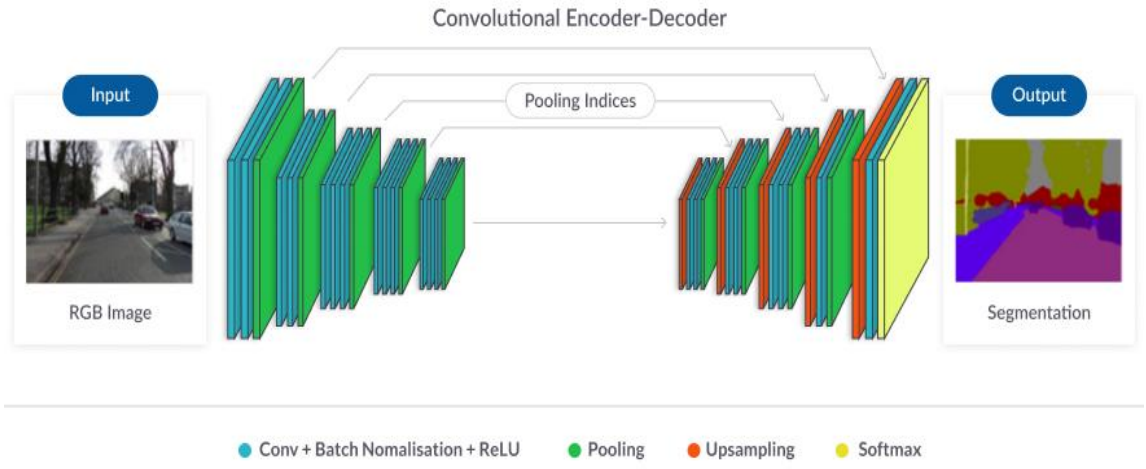


Nota. El gráfico muestra el ranking de modelos entrenados sobre el dataset COCO. Tomado de *Object Detection on COCO test-dev,* (paperswithcode, 2020).

- **Estimación de profundidad y Segmentación:** Para este tipo de tareas se utilizan redes denominadas *Fully Convolutional Network*. En la Figura 19 se ilustra la arquitectura de estas redes, formadas por un enconder y un decoder cuya función es comprimir la información y a partir de ello generar una nueva salida por pixel.

Figura 19

Arquitectura Fully Convolutional Network



Nota. El gráfico representa el modelo de red convolucional estructurada de forma de encoder y decoder. Tomado de *Image Segmentation in Deep Learning: Methods and Applications,* (missinglink.ai, s.f.).

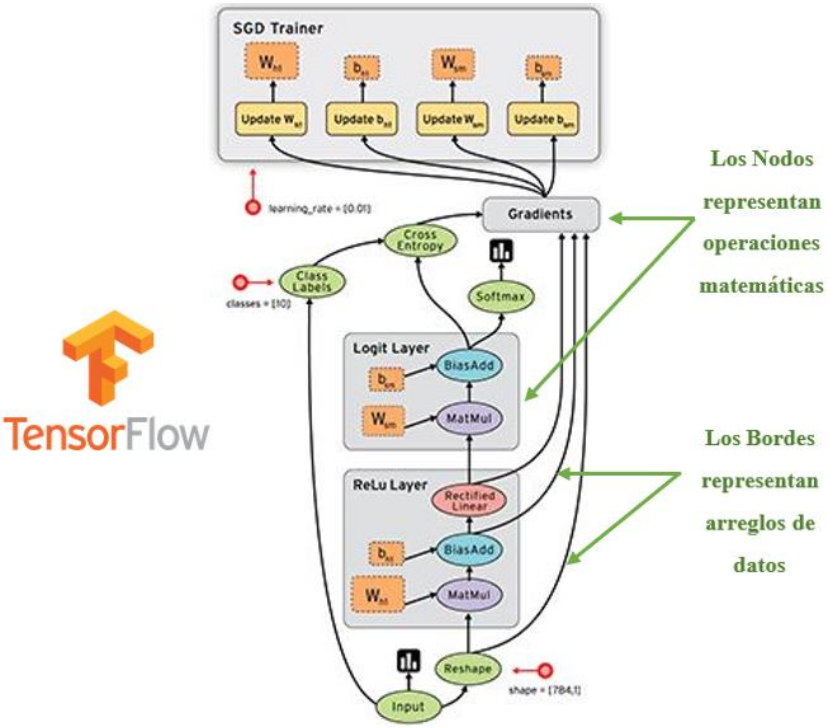
TensorFlow – Framework para machine learning

Tensorflow es una Plataforma de aprendizaje profundo de código abierto desarrollada por Google. Está conformada por librerías de computación matemática basadas en la manipulación, almacenamiento y transferencia de tensores (generalización del concepto de matrices) por medio de gráficos de flujo de datos, también llamados dirigidos como se ilustra en la Figura 20. Es muy utilizada para crear modelos de

aprendizaje automático e implementarlos de forma local o en la nube con aceleradores de hardware como *GPUs* y plataformas de computación paralela para procesamiento de tensor *CUDA*, además de implementaciones en navegadores o en dispositivos.

Figura 20

Representación de TensorFlow en gráficos dirigidos



Nota. El gráfico muestra el procesamiento matemático que realiza TensorFlow a través de grafos. Tomado de *Learn Tensorflow 4: Convolutional Neural Networks (CNNs),* (TensorFlow.org, 2019).

Tensorflow integra una API de aprendizaje profundo de alto nivel para construir, compilar y entrenar modelos, llamada *Keras*, que, con una interfaz amigable, modular y configurable, ofrece al usuario la facilidad de una rápida creación de prototipos. El uso de esta API mediante su implementación en *Python* como lenguaje principal, permitirá definir y entrenar redes neuronales.

Swift y *JavaScript* con *TensorFlow.js* son otros tipos de lenguajes que integra el ecosistema de Tensorflow que también apuesta por el edge computing con dispositivos integrados y móviles a través de la plataforma *TensorFlow Lite*. Adicionalmente disponen de *TensorFlow Extended (TFX)* para canalizaciones de producción.

Pre-Trained Model - Yolo v4

Su nombre proviene de sus siglas en inglés *(You Only Look Once)*. Es un modelo basado en el aprendizaje profundo de código abierto, bastante popular en la tarea de detección de objetos como se observa en la Figura 21. Su popularidad se debe a su robustez, lo cual genera resultados muy significativos en términos de velocidad y rendimiento. Por ello es considerada como una de las redes neuronales más rápidas y precisas en tiempo real en el conjunto de datos *COCO*, capaz de detectar hasta 80 tipos de objetos.

Figura 21

Detección de objetos con Yolo v4

Nota. El gráfico muestra los resultados de la inferencia del modelo YOLOv4.

A diferencia de otros sistemas que usan algoritmos basados en clasificación, caracterizados por ser lentos y computacionalmente costosos, *YOLO* utiliza algoritmos basados en regresión, que analizan toda la imagen, la divide en regiones para posteriormente predecir cuadros delimitadores y la clase a la que pertenece cada objeto con sus respectivas probabilidades o porcentajes de certeza.

Plataforma Cuda

El acrónimo *CUDA* proviene de sus siglas en inglés *(Compute Unified Device Architecture)*. Esta es una plataforma de computación paralela eficiente, común en las familias de *GPU Nvidia*, cuyo enfoque está en aprovechar el poder de procesamiento de estas para ejecutar miles de tareas simultáneamente, procesando gran cantidad de datos por ciclo de reloj, razón por la que es una plataforma muy empleada en el campo del *Deep Learning*.

Una *GPU Nvidia* internamente está conformada por cientos o miles de *CUDA Cores*, donde cada core es análogo a un core de *CPU*, pero menos sofisticados. En este punto la *CPU* actúa como un administrador y controlador de los datos en la PC, mientras que la *GPU* va ser la encargada del trabajo pesado, es decir de ejecutar simultáneamente gran cantidad de datos a través de los *CUDA Cores*.

Nvidia también ofrece la biblioteca de red neuronal profunda *Nvidia cuDNN*, que sirve como complemento junto con *CUDA* para la aceleración de *GPU*. Especialmente es utilizado para el entrenamiento de redes neuronales, centrándose en acelerar su proceso sobre diferentes frameworks de aprendizaje profundo.

Capítulo III

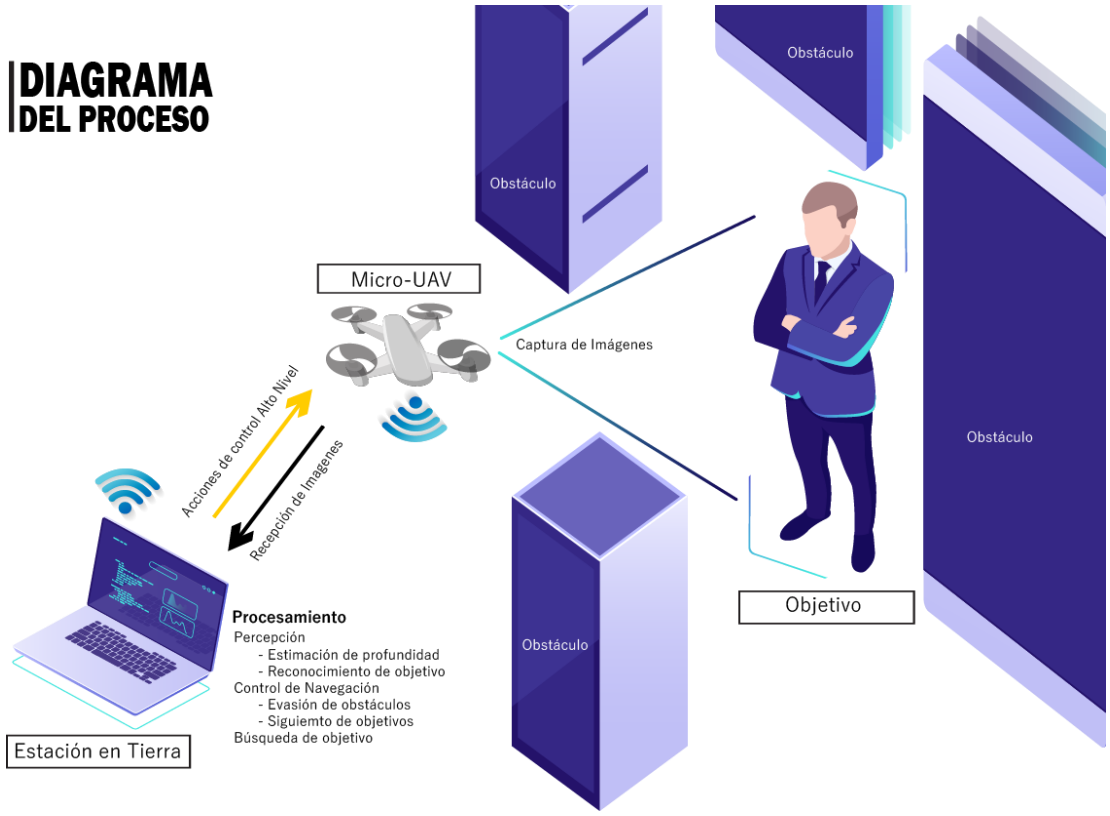
Descripción del Sistema

Especificaciones del Hardware del Sistema

El sistema consta de los siguientes componentes ilustrados en la Figura 22: un controlador o estación en tierra, un micro-UAV o estación aérea y el sistema de comunicación entre estos, siendo así parte de un sistema de aeronaves no tripuladas (UAS).

Figura 22

Esquema General del Sistema



Nota. El gráfico muestra un esquema general del funcionamiento del sistema en un entorno.

Estación en tierra

Está conformada de un ordenador portátil, quien va ser el encargado de manejar el software y varias herramientas con el fin de procesar los datos recibidos desde el *micro-UAV*, ejecutar una serie instrucciones y realizar las acciones de control de alto nivel. El ordenador debe contar con ciertas características que le permitirán tener un buen rendimiento al realizar tareas en tiempo real de alto costo computacional en el campo de *computer visión*. Como referencia, las características principales del ordenador usado son:

- Procesador: Intel Core i5-8300H CPU 2.30 GHz
- Memoria RAM: 8.00 GB
- Unidad de procesamiento gráfico: NVIDIA GeForce GTX 1050 Ti con 4GB GDDR5

Estación aérea (micro-uav)

El *micro-UAV* corresponde al cuadricóptero *Bebop 2* ilustrado en la Figura 23, fabricado por la empresa *Parrot*. Destinado a la recreación y también a su uso como una plataforma de desarrollo de código abierto.

Figura 23

Drone Parrot Bebop 2

Nota. El gráfico muestra el micro-UAV comercial usado.

Dotado de una autonomía de vuelo de 25 minutos y de una gama de sensores y tecnologías que le dan un increíble aerodinamismo, estabilidad y maniobrabilidad que lo hacen ser un diseño robusto. Además, es muy compacto y ligero a la vez, pesando tan solo 500 gr.

Especificaciones Generales

Diseñado en una configuración en cruz (x) con 4 motores potentes y hélices flexibles de eficiencia optimizada, con ello es capaz de volar incluso en condiciones extremas, con una capacidad de mantenerse estable en altitud y ante ráfagas de viento de hasta 64 km/h. Está equipado con un sistema de seguridad que detiene automáticamente las hélices cuando estas impactan alguna superficie y añade en su parte trasera un potente LED que permite su visibilidad a larga distancia. Mas detalles se muestran en la Tabla 2.

Tabla 2

Especificaciones técnicas Parrot Bebop 2

Especificaciones Generales	
Procesador	CPU dual-core con GPU quad-core
Almacenamiento	8 Gb de memoria flash
Sensores	Cámara vertical, ultrasónico, barómetro, acelerómetro, magnetómetro, giroscopio y GPS
Motores	4 motores brushless outrunner sin escobillas
Dimensiones	38 x 33 x 9 cm
Estructura	PA12 reforzada de Fibra de vidrio (20%) y Grilamid (casco)
Peso	500 gr
Batería	Li-Po: Ion de litio recargable de 2.700 mAh
Hélices	4 hélices de tres aspas
Cámara	14 Mp de resolución con lente tipo ojo de pez

Especificaciones Generales
Conectividad
Estándar Wi-Fi IEEE 802.11 a /b /g /n /ac
Red de banda dual MIMO con 2 antenas Wi-Fi dipolo dual de 2.4 y 5 GHZ
Potencia de transmisión hasta de 21 dBm
Alcance: 300 m con conexión a un smartphone/tablet y hasta 2 Km con mando RC Skycontroller

Nota. Esta tabla muestra los detalles técnicos de los componentes del micro-UAV Parrot Bebop 2. Tomado de *Parrot Bebop 2 FPV drone - Technical Specifications | Parrot Store Official*, (Parrot SA, 2020).

A continuación, se detalla las funciones de los sensores:

- Cámara de estabilización vertical: Usada para determinar la velocidad del *micro-UAV* a través de algoritmos de *Optical Flow*, haciendo una comparación entre capturas del suelo tomadas cada 16 milisegundos.
- Sensor Ultrasónico y de Barómetro: El primero realiza una medición de la altitud de vuelo hasta los 4,9 m, el segundo es el encargado de medir la presión del aire y a la vez mide la altitud desde los 4,9 m en adelante.
- Acelerómetro: Es el encargado de obtener la posición y las velocidades lineales en los 3 ejes de coordenadas.
- Magnetómetro: Conformado de tres ejes y una brújula incorporada para establecer la posición del *micro-UAV*.
- Giroscopio: Conformado de tres ejes encargados de medir la inclinación en pitch, roll y yaw.
- GPS + GLONASS: Es el chipset GNSS utilizado para geolocalización del *micro-UAV*.

Especificaciones de la Cámara

La cámara que lleva a bordo es uno de los puntos favorables del *micro-UAV*, por su calidad y estabilidad, que dan al usuario una gran experiencia en la toma fotos y grabación de videos en alta resolución. Tiene un campo de visión de 180°, pero se centra únicamente en una determinada sección de la imagen completa, debido a que las demás secciones están enfocadas en mantener la estabilización digital de video. Afortunadamente cuenta con un sistema que permite modificar digitalmente la orientación de esta y gracias a la ubicación de la cámara en el micro-UAV, permite llegar a capturar y filmar videos verticalmente. Mas detalles de sus especificaciones se muestran a continuación.

Tabla 3

Especificaciones técnicas de la cámara en imagen y video

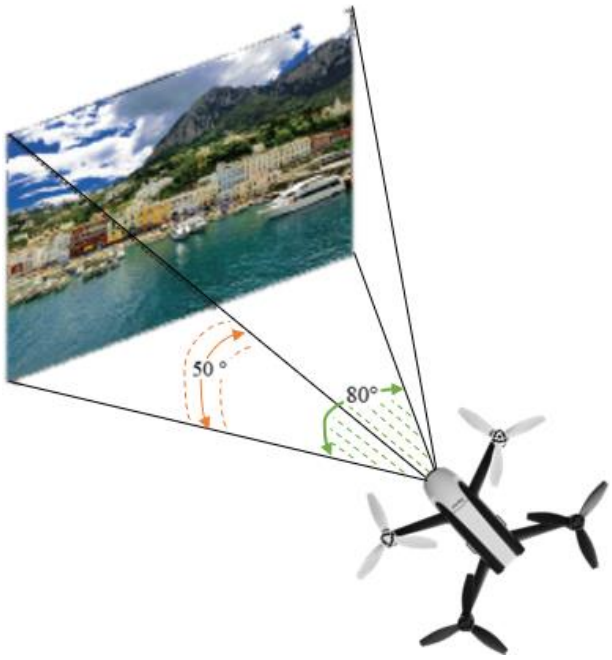
Imagen y Video
Cámara de 14 Mp sensor CMOS
Lente óptico tipo ojo de pez
Ángulo de visión de 180 grados con apertura de 1/2,3 “ (6 elementos de óptica)
Estabilización digital de video de 3 ejes
Resolución de la imagen de 4096 x 3072p
Resolución de video Full HD: de 1920 x 1080p a 30 fps
Formato de imagen: JPEG, RAW, DNG
Codificación de video H264

Nota. Esta tabla muestra los detalles técnicos de la cámara del micro-UAV Parrot Bebop 2. Tomado de *Parrot Bebop 2 FPV drone - Technical Specifications* | *Parrot Store Official*, (Parrot SA, 2020)

La sección enfocada por la cámara está limitada a un rango de visión como se observa en la Figura 24, que es de 50° en el eje vertical y de 80° en el eje horizontal.

Figura 24

Campo de visión en video del micro-uav Bebop 2



Nota. El gráfico muestra el ángulo de visión de la cámara del micro-UAV Parrot Bebop 2.

Desempeño

Los motores que permiten impulsar y dar maniobrabilidad al *micro-UAV*, determinan el rango de velocidades a manejar en los movimientos de desplazamiento y de rotación para su desempeño adecuado.

- ***Velocidad Horizontal:*** capaz de llegar a una máxima de 16 m/s, aproximadamente unos 60 km/h en tan solo 14 segundos
- ***Velocidad Vertical:*** su velocidad de ascenso va desde los 0.5 m/s hasta una máxima de 6 m/s, aproximadamente unos 21 km/h.
- ***Velocidad de rotación:*** va desde los 5 °/s hasta los 200 °/s

- ***Inclinación:*** durante el desplazamiento puede inclinarse desde un ángulo de 5° hasta los 35°.
- Además, su tiempo de frenado rodea los 4 segundos

Sistema de comunicación

El *micro-UAV Bebop 2* es quien se encarga de emitir una señal Wi-Fi de 2.4Ghz por defecto, actuando como servidor, cuya IP es 192.168.42.1. La estación en tierra en cambio actúa como cliente, estableciendo una conexión con la red emitida por el *micro-UAV* como se ilustra en la Figura 1. De esta manera el acceso a los datos emitidos desde el servidor y el envió de datos de control hacia él estarán basados en un intercambio de información *TCP* con el software detallado en el siguiente apartado.

Especificaciones del Software del sistema

El sistema operativo usado como base principal para el desarrollo del proyecto de investigación fue la distribución de *GNU/Linux Ubuntu 16.04 LTS* de código abierto*.* Se trata de una versión con soporte a largo plazo, donde prima la estabilidad. Es muy usado y preferido por desarrolladores por la libertad de hardware y software que esta ofrece, además de la estabilidad y compatibilidad al trabajar con varias herramientas.

Dentro del área de inteligencia artificial, este sistema es preferido por gran parte de los frameworks más conocidos como *TensorFlow*, *Pythorch*, *OpenCV*, *Keras*, entre otros. A ello se añade el soporte a tarjetas gráficas para mayor poder computacional y las herramientas para el control del *micro-UAV*, donde su marco de desarrollo está basado en sistemas *Linux.*

Entorno Robot Operating Systems

Se trata de un Meta-sistema operativo de código libre, que impulsa la investigación y el desarrollo de la robótica de forma colaborativa, teniendo a Ubuntu como su principal plataforma de soporte. Su enfoque es el proporcionar servicios para trabajar con elementos de hardware, drivers y controladores de dispositivos vinculados, servicios de transmisión de mensajes entre procesos, además de un conjunto de herramientas y bibliotecas para desarrollo, visualización y administración. Es aplicado en la robótica en diversas plataformas, como los son brazos robóticos, bases móviles, drones y otro tipo de estructuras robóticas complejas, descentralizando todo el proceso y simplificando su programación mediante una serie de tareas más sencillas interconectadas entre sí, compiladas en distintos lenguajes tales como *C++, Python*, *Lisp*, entre otros.

Específicamente en el desarrollo de este proyecto de investigación se usó la versión *ROS Kinetic*, dirigida para trabajar sobre sistema operativo *Ubuntu 16.04 (Xenial).*

Estructura a nivel de grafos de computación

Su funcionamiento está encaminado por una arquitectura de grafos de computación que representa la red de procesos que realiza *ROS*. Esta red lo conforman elementos como “nodos”, que pueden tener una comunicación externa, enfocada a realizar tareas de recepción de datos proveniente de sensores o una comunicación internar con otros nodos que se encargan de tareas de procesamiento.

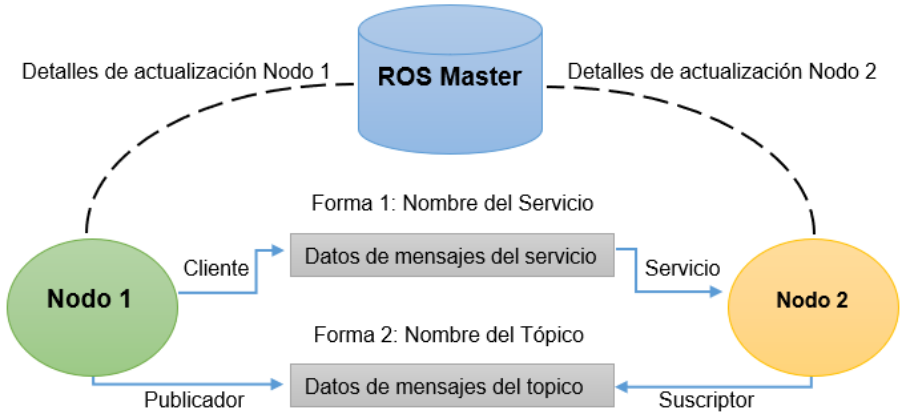
La comunicación entre nodos puede ser de dos formas, la primera en base a “servicios” de comunicación punto a punto cuyos “mensajes” son leídos únicamente por los extremos de esa comunicación y la segunda es a través de suscripción/publicador,

que actúa como una especie de broadcast donde todos aquellos elementos que estén escuchando esa comunicación recibirán los mensajes.

La vía por donde se establece esta comunicación suscripción/publicador se denominan “tópicos”, cada uno con un nombre especifico referente al contenido del mensaje que está manejando. Estos tópicos se encargarán de recibir y enviar mensajes de un tipo de dato específico desde y hacia los nodos que estén vinculados a él. Con ello un nodo puede publicar o suscribirse a los tópicos de su interés. En la Figura 25 se muestra un diagrama de este proceso, donde la pieza fundamental es el *ROS Master* que es quien se encarga de administrar, registrar y organizar los mensajes y servicios entre los nodos, permitiéndoles que estos puedan comunicarse entre sí.

Figura 25

Comunicación interna entre nodos



Nota. El gráfico muestra un esquema del proceso de comunicación y envió de mensajes en ROS.

Todos estos conceptos mencionados son los necesarios para el desarrollo de este proyecto, no obstante *ROS* abarca muchos más que pueden ser revisados a detalle en (Garcia J. , 2016).

Comunicación entre Ros y el micro-uav Bebop 2

ROS cuenta con una variedad de librerías que utilizan su sistema de comunicación para interactuar con distintas plataformas robóticas, una de ellas es el controlador *bebop*_autonomy de código abierto, basado en el SDK oficial de la compañía Parrot llamado *ARDroneSDK3* y desarrollado para los modelos de cuadricópteros *Bebop 1* y *Bebop 2*. Este controlador permite obtener y manipular la información que el micro-UAV publica a través de tópicos y a la vez enviar señales de control hacia él desde un ordenador.

Para iniciar la comunicación el ordenador tiene que establecer una conexión a la red Wi-Fi emitida por el *micro-UAV*, hecho esto se añade el espacio de trabajo de bebop a través de la siguiente línea de comando digita en el terminal de *Ubuntu:*

```
$ source ~/bebop_ws/devel/setup.[bash|zsh]
```

Lo siguiente es ejecutar el archivo “launch”:

```
$ roslaunch bebop_driver bebop_node.launch
```

Este archivo levanta el *bebop_node* con una serie de tópicos y además ejecuta algunas otras tareas de configuración:

- Establece la IP que tendrá el *micro-UAV*, por defecto es 192.168.42.1
- Ejecuta el archivo *defaults.yaml* que contiene una configuración por defecto de parámetros de pilotaje, movimiento, red, imagen y GPS, que se encuentran a mayor detalle en (Grijalva, 2019).
- Ejecuta el archivo de calibración de la cámara *bebop2.camera.calib.yaml*

Se mantiene la configuración por defecto de la mayoría de los parámetros del archivo *defaults.yaml*. En la Tabla 4 se muestran los únicos parámetros modificados:

Tabla 4

Parámetros configurados de pilotaje, velocidad e imagen

Descripción	Parámetro	Valor	
		Por defecto	**Configurado**
Inclinación máxima para pitch y roll	MaxTiltCurrent	5 °	20 °
Velocidad máxima de rotación en yaw	MaxRotationSpeedCurrent	13 °/s	100 °/s
Velocidad máxima de rotación en pitch y roll	MaxPitchRollRotationSeepdCurrent	80 °/s	100 °/s
Resolución del streaming y grabación de video	VideoResolutionsType	0	1

Nota. Esta tabla muestra las configuraciones de vuelo realizadas en el micro-UAV.

La configuración de los tres primeros parámetros de velocidad de la Tabla 4 tienen como objetivo mejorar la respuesta en el desplazamiento del *micro-UAV* tanto en trayectorias rectas y en curvas cerradas a una velocidad de 1 m/s, que se aproxima a la velocidad de promedio de caminata de una persona. El último parámetro establece una resolución de 856 x 480 pixeles en streaming de video, adecuado para un rápido procesamiento y una rápida tasa de transmisión de las imágenes de aproximadamente 30 Hz equivalente a unos 30 FPS. El parámetro de altura no se configura, se mantiene por defecto para que realice el vuelo a una altura fija de 1 metro.

Se elaboraron scripts bajo el lenguaje de programación *Python*, uno de estos implementa la clase “bebop2” con algoritmo de comunicación con el núcleo de *ROS* y el *micro-UAV* a través del controlador *bebop_autonomy.* Entre los guiones de este script se encuentran las instrucciones para publicar o suscribirse a tópicos que maneja el *micro-*

UAV. En los siguientes apartados se indican los únicos tópicos usados de un abanico de tópicos disponibles, que se listan a través del comando:

```
$ rostopic list
```

Principales tópicos publicados

En la Tabla 5 se detalla los tópicos publicados que se usaron para controlar los movimientos del *micro-UAV* y de su cámara. Estos pueden ser escritos en lenguaje *Python* mediante la biblioteca *Rospy* e invocando a una de sus funciones llamada *publish().*

Tabla 5

Lista de tópicos publicados que se usaron

Descripción	Publicador	Tópico	Mensaje
Realiza la acción de despegue	Pub_Land	/bebop/takeoff	std_msgs.msg.Empty
Realiza la acción de aterrizaje	Pub_Takeoff	/bebop/land	std_msgs.msg.Empty
Realiza los movimientos lineales en x, y, z	Pub_Twist	/bebop/cmd_vel	geometry_msgs.msg.Twist
Realiza el movimiento virtual de la cámara	Pub_CamTwist	/bebop/camera_control	geometry_msgs.msg.Twist
Detiene los motores	Pub_Reset	/bebop/reset	std_msgs.msg.Empty

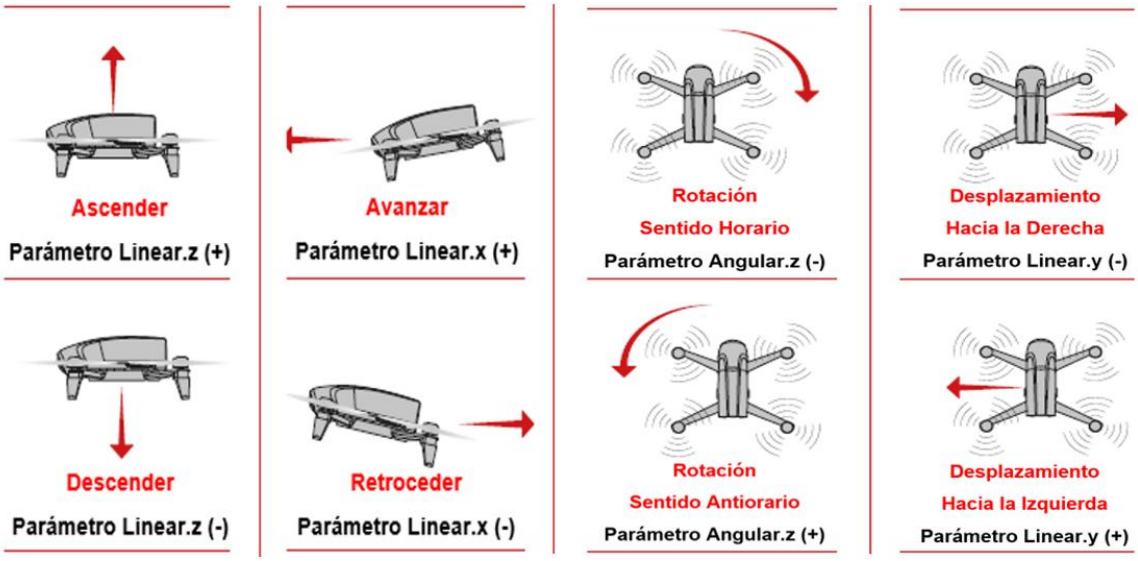
Nota. Esta tabla muestra la lista de tópicos usados para realizar distintas acciones de control en el micro-UAV.

El tópico “/bebop/camera_control” permite configurar el área de visualización de la cámara, a través de una variación angular de su posición digitalmente que va desde [-80 ° ⋯ 80°] en forma vertical. La configuración requerida para la cámara es que enfoque hacia al centro, por lo tanto, el valor publicado que permite esto es el de 0°.

En la Figura 26 se ilustran los movimientos lineales y angulares en los distintos ejes del *micro-UAV*. Para realizar estas acciones de movimiento se utiliza el tópico "/bebop/cmd_vel" con los parámetros "linear.x", "linear. y" y "linear.z", dentro del rango de acción que va desde [-1 ⋯ 1], siendo la velocidad mínima el valor de 0. Tales valores se enviarán en el mensaje a publicar, dando prioridad a ejecutar el último mensaje recibido. No obstante, el movimiento "linear.z" se mantendrá en 0 para este proyecto, es decir no se manipula la elevación del *micro-UAV*, solo se trabaja como los movimientos en los ejes paralelos al terreno que son los ejes "x" e "y". El eje "z" pasa a ser únicamente usado para movimientos de rotación a través del parámetro "angular.z".

Figura 26

Movimientos generados por los parámetros de control del micro-uav



Nota. El gráfico muestra los movimientos permitidos en el micro-UAV Parrot Bebop 2.

Principales tópicos suscritos

En la Tabla 6 se detalla algunos de los tópicos suscritos que se usaron para adquirir las imágenes de la cámara y conocer el estado de la batería del *micro-UAV*. Estos pueden también ser escritos en lenguaje *Python* mediante la biblioteca *Rospy*.

Tabla 6

Lista de tópicos suscritos que se usaron

Descripción	Suscriptor	Tópico	Mensaje
Estado de la batería	Sub_Battery	/bebop/states/common /CommonState /BatteryStateChanged	CommonCommonState BatteryStateChange
Obtención de imagenes	Sub_Image	/bebop/image_raw	sensor_msgs.msg.Image

Nota. Esta tabla muestra la lista de tópicos suscritos que se usaron para obtener datos desde el micro-UAV.

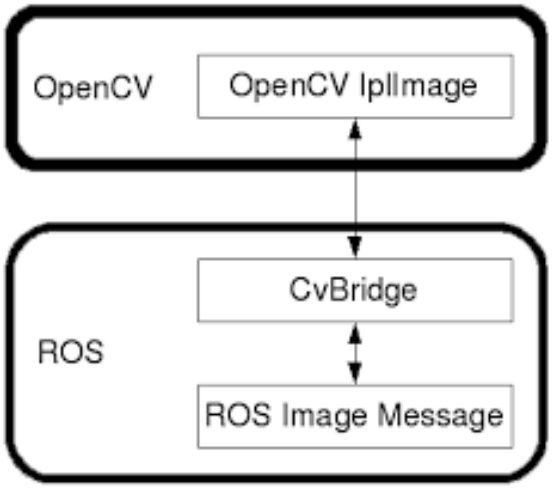
El tópico que permite conocer el estado de la batería es usado para llevar un monitoreo del nivel de cargar mientras se realiza el vuelo para así prever cuando el *micro-UAV* debe abortar tal misión.

En el esquema de la Figura 27 se muestra el proceso de obtención de las imágenes del micro-UAV que son llevadas a cabo por el suscriptor de imágenes una vez levantado el nodo que establece la comunicación con *ROS*. El mensaje de la imagen es publicado constantemente en el tópico “/bebop/image_raw, posteriormente se realiza un llamado a la función *callback()*, encargada de ejecutar un procesamiento intermedio, basado en una codificación del mensaje de *ROS* a un objeto *OpenCV*, mediante la biblioteca CvBridge, disponible en *ROS* y finalmente pasa ser almacenada en el parámetro “Image”.

Las imágenes recibidas están en base a los parámetros configuración en la Tabla 4, tendrán una resolución 856 x 480 pixeles a una tasa de 30 Hz.

Figura 27

Conversión de Imagen de mensaje Ros a objeto OpenCv



Nota. El gráfico muestra el proceso que conlleva la obtención de una imagen en ROS a través de OpenCV.

Existen otros tópicos que permiten suscribirse de ser necesario. Estos envían datos inerciales del *micro-UAV,* pero lo hacen a una frecuencia de 5 Hz, que puede ser una limitante al momento de realizar modelación a partir de ellos.

Capítulo IV

Detección, seguimiento de persona y evasión de obstáculos

El UAV precisa del conocimiento de su entorno para interactuar frente a distintas situaciones, de manera que necesita percibir los elementos presentes para así ajustar su comportamiento frente a ellos. Esto es posible gracias una variedad de sensores que llevan abordo comúnmente los UAVs, entre uno de los más importantes es su cámara incorporada, dando apertura así a un abanico de soluciones utilizando visión por computador, como detectar objetos, seguirlos y de paso estimar profundidad para evadir obstáculos presentes en ese entorno.

Para el desarrollo de estas soluciones fue necesario el uso de las siguientes bibliotecas y frameworks con sus respectivas versiones:

- Python v2.7
- OpenCV v3.3.1
- Driver Nvidia v384.1
- TensorFlow v1.12
- CUDA v9.0.176
- cuDNN v7.3
- YOLO v4

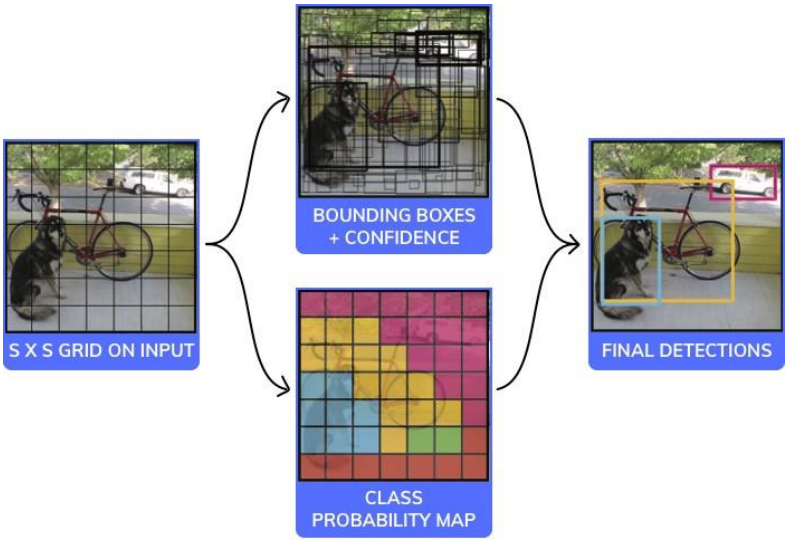
Detección de objetos con la red pre-entrenada Yolo v4

La detección de objetos es una de las tareas más populares de aprendizaje profundo en el campo de *computer visión*. Tiene la facultad de encontrar distintos objetos en una imagen, localizarlos con cuadros delimitadores cuyas coordenadas son pixeles e

indicar una respectiva etiqueta sobre la clase en particular a la que pertenece. En esta tarea se distinguen dos técnicas importantes:

Figura 28

Clasificación, localización en la detección de objetos



Nota. El gráfico muestra las técnicas usadas en la tarea de detección de objetos. Tomado de *How to easily Detect Objects with Deep Learning on Raspberry Pi | Nanonets, (Jain, 2018)*.

- **Clasificación de objetos:** se encarga de asignar una etiqueta a una imagen en particular a partir de la predicción de la probabilidad de que pertenezca a una clase especifica como persona, auto, perro, gato, silla, etc. Se basa principalmente en las características de la imagen.
- **Localización de objetos:** predice la probabilidad de que un objeto existe en la imagen acompañado de su ubicación dentro de ella mediante la asignación de un cuadro delimitador que va acompañado con la etiqueta predicha en la etapa de clasificación.

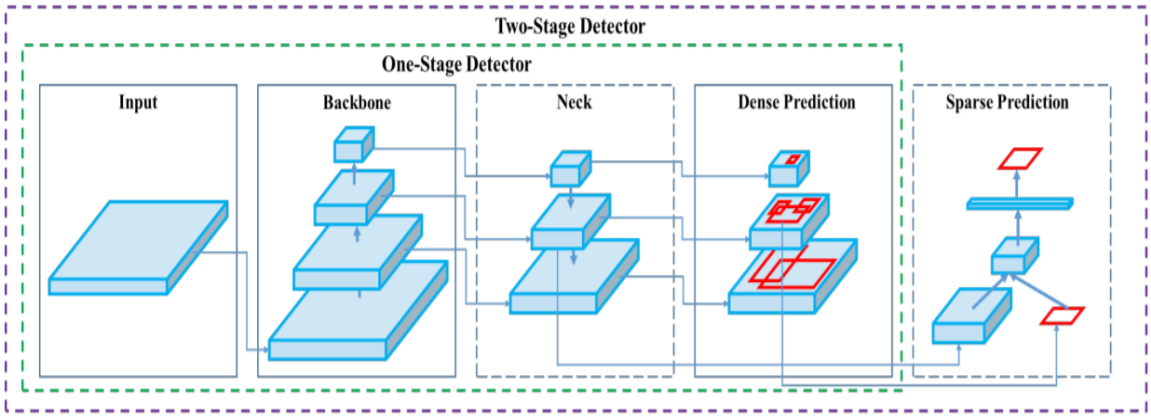
La red pre-entrenada *YOLOv4* es una versión mejorada de su antecesora *YOLOv3*. Sus principales cambios fueron en ciertos componentes dentro de su arquitectura.

Arquitectura del detector de objetos

En la Figura 29 se muestran los elementos que componen esta arquitectura, la sección One-Stage Detector (encerrada en líneas verdes) es la correspondiente a la arquitectura de *YOLOv4*. En el caso de redes como Fast R-CNN y Mask R-CNN añaden la etapa de Sparse Prediction, representada por la sección Two-Stage Detector (encerrado en líneas violetas) (Bochkovskiy, Wang, & Liao, 2020).

Figura 29

Arquitectura del detector de objetos



Nota. El gráfico muestra los componentes que conforman la arquitectura YOLOv4. Tomado de *YOLOv4: Optimal Speed and Accuracy of Object Detection,* (Bochkovskiy, Wang, & Liao, 2020).

- **Input (entrada):** este bloque representa el ingreso de la imagen

- **Backbone (Columna vertebral):** representa la red neuronal *CSPDarknet53* encargada de extraer un mapa de características de la imagen de entrada.
- **Neck (cuello):** es un subconjunto del *Backbone* y está compuesto de varias rutas de arriba hacia abajo y viceversa que dan mayor robustez y capacidad de discriminación. Usa el bloque *SPP* (Spatial pyramid pooling), *PANet* (Path aggregation Network), *Mish activation, CSP* (Cross-stage partial connections) y *MiWRC* (Multiinput weigted residual connections)
- **Head (cabeza):** *Dense Predictions*, es el bloque encargado de las predicciones, está comandado por el detector de predicciones densas de *YOLOv3***.** Adicionalmente cuenta con *Mish activation, SPP-block, SAM-block, PAN path-aggregation block* y *DIoU-NMS.*

Modelo pre-entrenado

El modelo fue entrenado por los autores (Bochkovskiy, Wang, & Liao, 2020) sobre el conjunto de datos *COCO* que cuenta con 80 clases de objetos a detectar, entre ellos están personas. Este entrenamiento lo realizaron con una *GPU NVIDIA 2080 TI* teniendo en cuenta los siguiente hiperparámetros:

- 500.500 épocas o pasos de entrenamiento
- Una taza de aprendizaje de 0.01 con la estrategia de decadencia de pasos.
- Una vez llegado a los 400.000 pasos, el factor de aprendizaje fue multiplicado por 0.1 y en los 450.000 pasos nuevamente se volvió a multiplicar por 0.1.
- El *momentum* fue de 0.9
- Pérdida de peso de 0.005

El modelo generado por el entrenamiento puede variar su rendimiento en precisión y velocidad dependiendo del tamaño (width y height) con el cual se desea que ingrese la imagen a esta red pre-entrenada para la inferencia, como se muestra en la Tabla 7.

Las evaluaciones del modelo entrenado fueron hechas en una *GPU NVIDIA: RTX 2070, Tesla V100* y se añade además las pruebas realizadas en el ordenador usado para la *Estación en Tierra*, que cuenta con una *GPU* de menos capacidades.

Tabla 7

Performance del modelo pre-entrenado Yolo v4

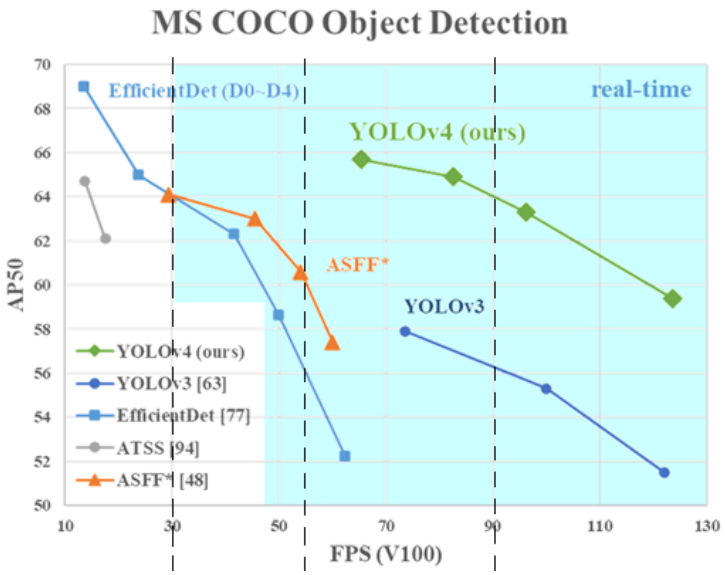
Modelos	**mAP**	**FPS**			**FLOPS**
		RTX 2070	**Tesla V100**	**Estación en Tierra GTX 1050 Ti**	
YOLOv4-608 width=608 y height=608	65.7%	34	62	9	128.5 BFlops
YOLOv4-512 width=512 y height=512	64.9%	45	83	12	91.1 BFlops
YOLOv4-416 width=416 y height=416	62.8%	55	96	16	60.1 BFlops
YOLOv4-320 width=320 y height=320	60.0%	63	123	20	35.5 BFlops
YOLOv4-tiny	40.2%	330	-	-	6.9 BFlops

Nota. Esta tabla muestra los valores del rendimiento del modelo pre-entrenado en distintas GPUs, incluyendo la disponible en la estación en tierra de este proyecto de investigación.

La forma de medir la precisión de los detectores de objetos es a través de la métrica *mAP* (mean average precision) de la Tabla 7. Se la obtiene a partir de algunos valores, entre los cuales está la medida de que tan cercano es la predicción a la verdad fundamental. El objetivo es que cada modelo indique una medida estimada de cuan precisa son las predicciones, es decir que porcentaje de ellas son correctas.

Figura 30

Velocidad y precisión de Yolo v4 en gpu Nvidia Tesla V100



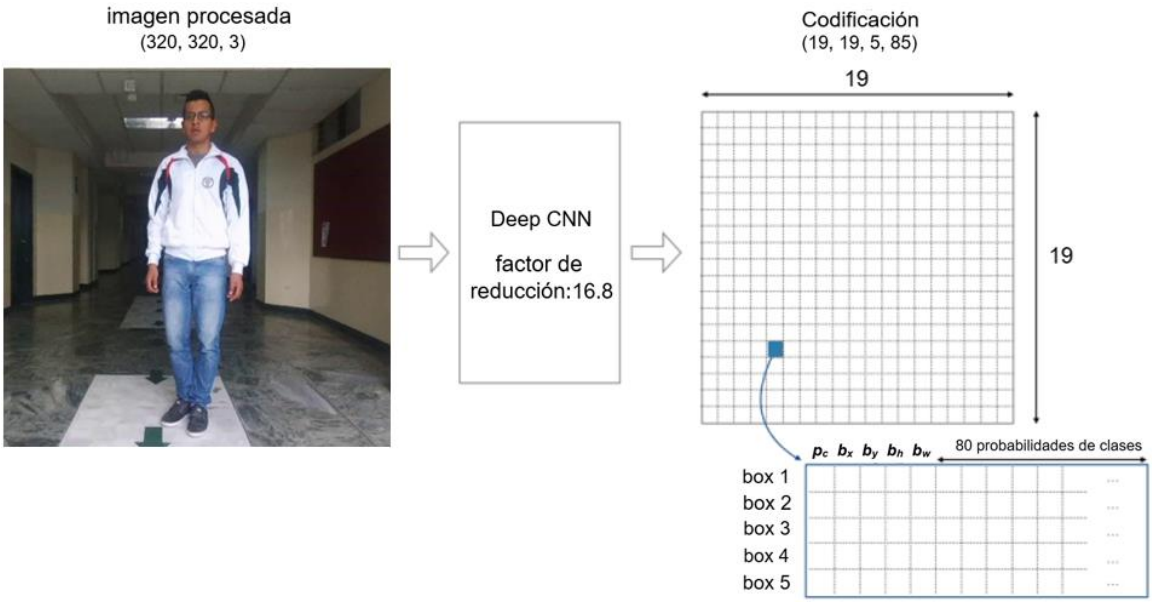
Nota. El gráfico muestra el rendimiento del modelo *YOLOv4* evaluado frente a otros modelos de distintas arquitecturas existentes en el estado del arte de la tarea de detección de objetos sobre el dataset COCO. Tomado de *YOLOv4: Optimal Speed and Accuracy of Object Detection,* (Bochkovskiy, Wang, & Liao, 2020).

Funcionamiento de Yolo v4

YOLO es una red totalmente convolucional, detecta múltiples objetos en una imagen y es responsable de predecir un cuadro delimitador para cada objeto encontrado, la clase a la que pertenece y su probabilidad o porcentaje de certeza. Como se observa en la Figura 31, el proceso de la red consiste en dividir la imagen en una celda de 19 x 19, cada celda se la denomina núcleo de detección en mapas de características y son las encargadas de predecir cada una cinco cuadros delimitadores sobre las secciones que estiman que se encuentre un objeto.

Figura 31

Proceso de predicción de cuadros delimitadores



Nota. El gráfico muestra el método que usa el modelo *YOLOv4* para analizar una imagen en busca de objetos.

Cada *frame* o imagen procesada e interpretada contendrá cientos de cuadros delimitadores y algunos tal vez no contengan ningún objeto. La solución es desechar la mayor cantidad de cuadros a través de un proceso *supresión no máxima*, que actúa como una especie de filtrado en función de la probabilidad de la clase del objeto, para así finalmente obtener como resultado de la inferencia un único cuadro delimitador para cada objeto, como se observa en la Figura 32. Durante este proceso es importante considerar los parámetros que conforman un cuadro delimitador para posteriores acciones dentro del proyecto.

Figura 32

Proceso de eliminación de cuadros delimitadores excesivos

Nota. El gráfico muestra como los cuadros delimitadores son eliminados, manteniendo el cuadro más preciso que encierre completamente al objeto detectado.

Donde:

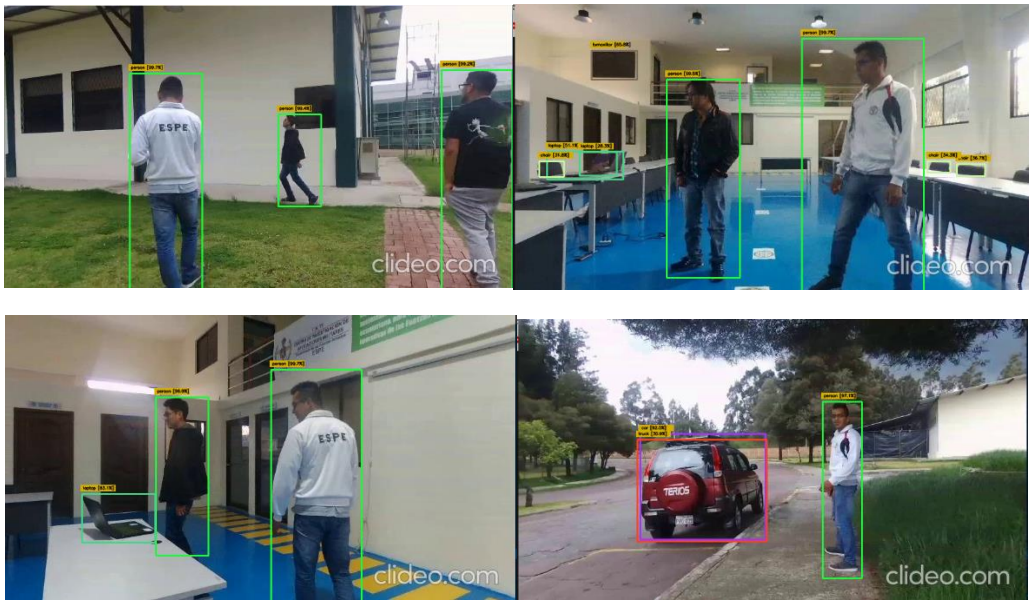
- p_c: es la probabilidad de la clase de objeto
- (b_x, b_y): representan la posición central del cuadro delimitador en pixeles.
- b_w: es el ancho del cuadro delimitador
- b_h: es el alto del cuadro delimitador
- c: indica la clase a la que pertenece el objeto

Una vez definido el funcionamiento, el siguiente paso a realizar sobre el modelo pre-entranado fue configurarlo para trabajar con un tamaño de *width=320* y *height=320*, en vista de que es el modelo que otorga los mejores resultados en precisión y velocidad para la *Estación en Tierra* que se dispone, como se indica en la Tabla 7. Con respecto a la velocidad, se sabe que el rendimiento que se obtiene es relativo a las demás *GPUs*, aproximadamente la inferencia se ejecuta a unas tres veces menos que en una *GPU NVIDIA RTX 2070*, es decir a unos 20 FPS. Pero aun así esta configuración posibilita las

inferencias a una velocidad adecuada para un procesamiento cercano a tiempo real sin sacrificar demasiado la precisión.

Figura 33

Detección de objetos con Yolo v4 implementado en la Estación en Tierra



Nota. El gráfico muestra las inferencias de la detección de objetos en distintos escenarios.

Posteriormente el algoritmo fue adaptado para reconocer únicamente personas, que es el objetivo de este proyecto de investigación.

Seguimiento de objetos en una imagen

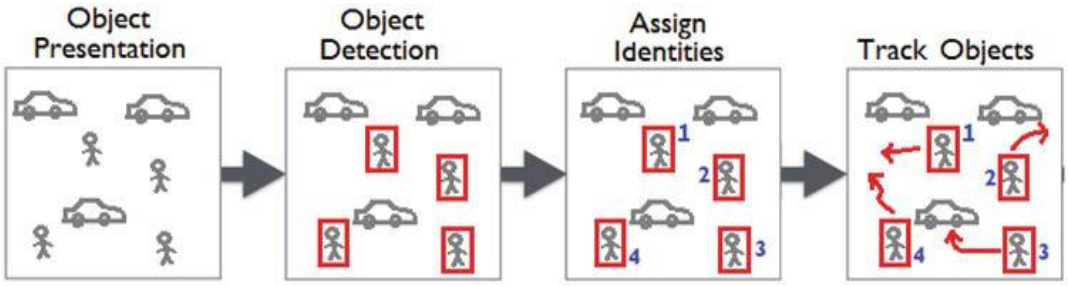
Aborda la tarea de localizar múltiples objetos en movimiento y a la vez rastrearlos en los cuadros posteriores de un video (streaming o pregrabado) a lo largo del tiempo, lo que implica la correspondencia cada objeto entre un cuadro actual y el posterior. Esta técnica asiste tanto en la identificación como en el rastreo sobre cada persona detectada, además ayuda a liberar la carga computacional generada por el detector en cada *frame*, de modo que la detección únicamente se ejecutará en determinados *frames*, mientras

que el seguidor que requiere menos procesamiento, se encargará de predecir las detecciones en los *frames* omitidos por el detector.

Este enfoque es conocido generalmente como sistemas *MOT* (Multi-Object Tracking) o seguimiento de múltiples objetos como se representa en la Figura 34. Su columna vertebral es un detector de objetos cuyas detecciones son posteriormente enviadas a un rastreador o sistema *MOT* que es quien se encarga de realizar el seguimiento a partir de la apariencia y características espacio-temporales del objeto en movimiento. En función de ello a cada objeto se le asigna una única identidad siempre y cuando permanezca dentro del cuadro de la imagen.

Figura 34

Pasos involucrados en un sistema Multi-Object Tracking



Nota. El gráfico representa en forma resumida el proceso de un sistema MOT. Tomado de *Visual Object Tracking with Deep Neural Networks*, (Khan, Tariq, & Ghani Khan, 2019).

Sistemas de este tipo son muy usados en cámaras de video vigilancia, en cámaras de control de tráfico, imágenes médicas, entre otros, con el fin de rastrear el movimiento individual de cada objeto, analizar su trayectoria, su comportamiento, etc. En (Khan, Tariq, & Ghani Khan, 2019) se detallan una variedad de sistemas *MOT*, con sus respectivos resultados de rendimiento, en función de las evaluaciones realizadas sobre un conjunto de datos provistos en la competencia *MOTChallenge*.

Tanto técnicas tradicionales de *computer vision* como técnicas de aprendizaje profundo, forman parte de estos sistemas, de manera que los hacen más robustos y a la vez se vuelven más costosos computacionalmente. Por este último motivo, se consideró para el desarrollo de este proyecto de investigación únicamente tomar como base las técnicas tradicionales usadas por estos y el proceso que involucra en la Figura 34, mas no algún sistema en particular.

Filtro de Kalman y Algoritmo Húngaro

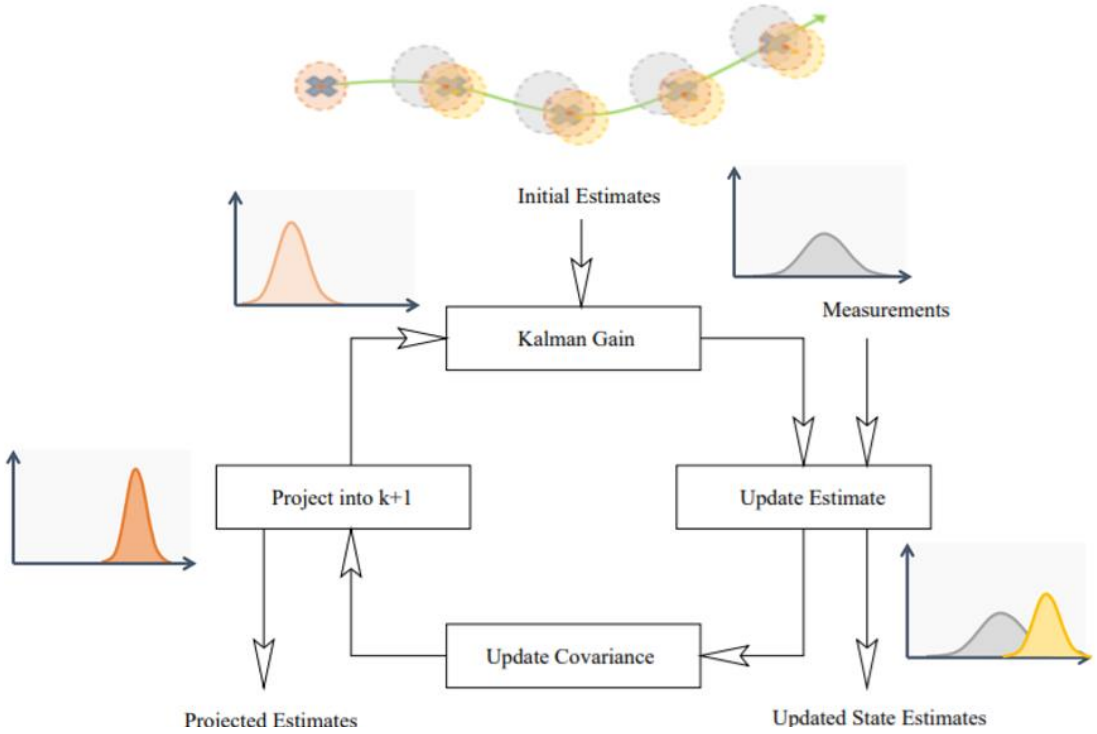
Estas dos técnicas tomadas para implementar un seguidor de objetos forman parte de algunos de los sistemas *MOT*. Entre los más conocidos están el sistema *SORT* y *Deep SORT*, que emplean en su algoritmo un *Filtro de Kalman* para sus marcos de entrada, seguido de un *algoritmo Húngaro* que les permite hallar asociaciones en pistas visuales (Khan, Tariq, & Ghani Khan, 2019).

El *Filtro de Kalman* es considerado una solución óptima en procesos de seguimiento y predicción de datos lineales o no lineales (Lacey, 2015). Su función es actuar de forma recursiva como se muestra en la Figura 35, toma las detecciones previas (con el ruido de proceso[1] y medición[2] involucrados) hechas por el detector *YOLOv4,* generando nuevas predicciones de hacia donde se mueve el objeto. La siguiente tarea es asociar estas predicciones con las nuevas detecciones, y es aquí donde entra en juego el simple y efectivo método orientado para resolver problemas de asociación, llamado *algoritmo Húngaro*.

[1] Ruido de proceso: es el ruido generado por el modelo en la predicción.
[2] Ruido de medición: es el ruido proveniente de las detecciones, debido a cambios de luminosidad, cambios de velocidad y movimientos bruscos del objeto o de la cámara al estar montada sobre un micro-UAV en movimiento.

Figura 35

Resumen del algoritmo recursivo del Filtro de Kalman Extendido



Nota. El gráfico representa el proceso recursivo que sigue el Filtro de Kalman Extendido, útil para aplicativos con comportamientos poco predecibles, caóticos, en fin, sistemas no lineales. Tomado de *Tutorial: The Kalman Filter | MIT - Massachusetts Institute of Technology*, (Lacey, 2015).

1. El proceso comienza con un estado inicial del objeto con una incertidumbre inicial representado por la gaussiana y el círculo rojo, cuya posición inicial esta simbolizada por la “x” azul. De aquí surge una primera predicción (simbolizado por la gaussiana y la circunferencia ploma) junto con un bloque *Kalman Gain* que especifica la confianza de la predicción frente a la medición.

$$K_k = P'_k H^T (H P'_k H^T + R)^{-1} \quad (4)$$

Donde:

- K_k: constante de normalización
- P_k: error de la matriz de covarianza
- H: conexión entre el vector de estados y el vector de medición
- R: covarianza del modelo de ruido asociado al error de medición

2. La siguiente etapa se lleva a cabo en el bloque *Update Estimate*. Este proceso de actualización comienza con la ejecución de una medición del ruido proveniente del detector juntamente con una corrección para encontrar la medida pronosticada (simbolizada por la gaussiana y la circunferencia dorada).

$$\hat{x}_k = \hat{x}'_k + K_k(z_k - H\hat{x}'_k) \tag{5}$$

Donde:

- x_k: vector de estado del proceso
- z_k: medición actual de la señal de información

3. Lo anterior da lugar a la actualización del estado de la matriz de covarianza en el bloque *Update Covariance.*

$$P_k = (I - K_k H)P'_k \tag{6}$$

Donde:

- I: matriz identidad

4. El resultado se presenta en el bloque *Project* que tendrá la posición actualizada del objeto, lo que da paso a un nuevo proceso de predicción. Todo el proceso se irá alimentado en cada interacción de datos como la velocidad, posición y dirección del movimiento del objeto, y como se observa en la Figura 35, este

puede reducirse a una propagación y actualización de gaussianos junto con la actualización de sus covarianzas.

$$\hat{x}'_k = \Phi \hat{x}_k \tag{7}$$

$$P_{k+1} = \Phi \mathrm{P_k} \Phi^T + Q \tag{8}$$

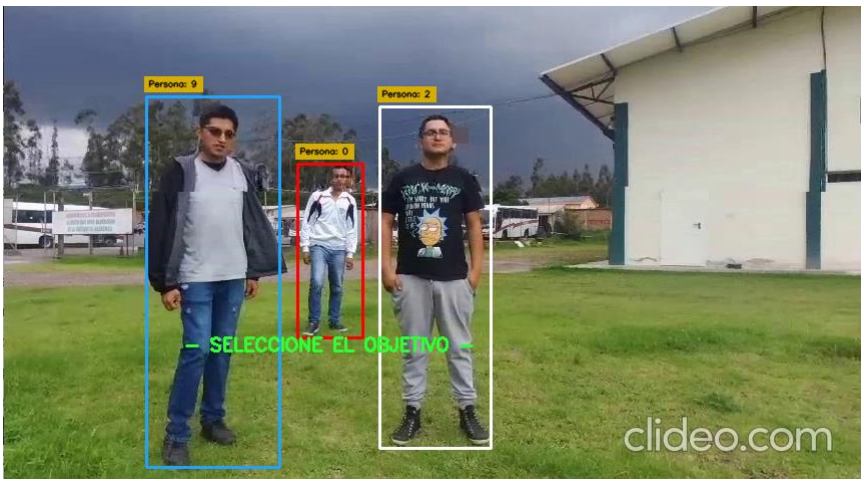
Donde:

- Φ: matriz de transición de estados
- Q: covarianza del modelo asociado al ruido blanco del proceso

Si bien es cierto el *Filtro de Kalman* tiene un mejor desempeño en sistemas lineales que conlleven procesos gaussianos. Pero en el caso de un seguidor pueden llegar a darse no linealidades debido a movimiento bruscos y poco predecibles, no obstante, la mayor cantidad de procesos llegan a ser parte del reino gaussiano (Jain, 2018).

Figura 36

Seguidor de Múltiples Personas



Nota. El gráfico muestra el resultado de la ejecución del algoritmo de seguimiento de personas de (Menon & Somawat, 2018) junto con el rastreador, que es alimentado con

las detecciones hechas previamente por el detector *YOLOv4*. En la ejecución el rastreador se encarga de asignar a cada persona automáticamente un único Id.

Una vez que se ha detectado a cada una de las personas, el usuario señala a cuál de ellas desea seguir, lo que lleva a un proceso de discriminación como se ilustra en la Figura 37, en el que el algoritmo se centra únicamente en el seguimiento de la persona seleccionada.

Figura 37

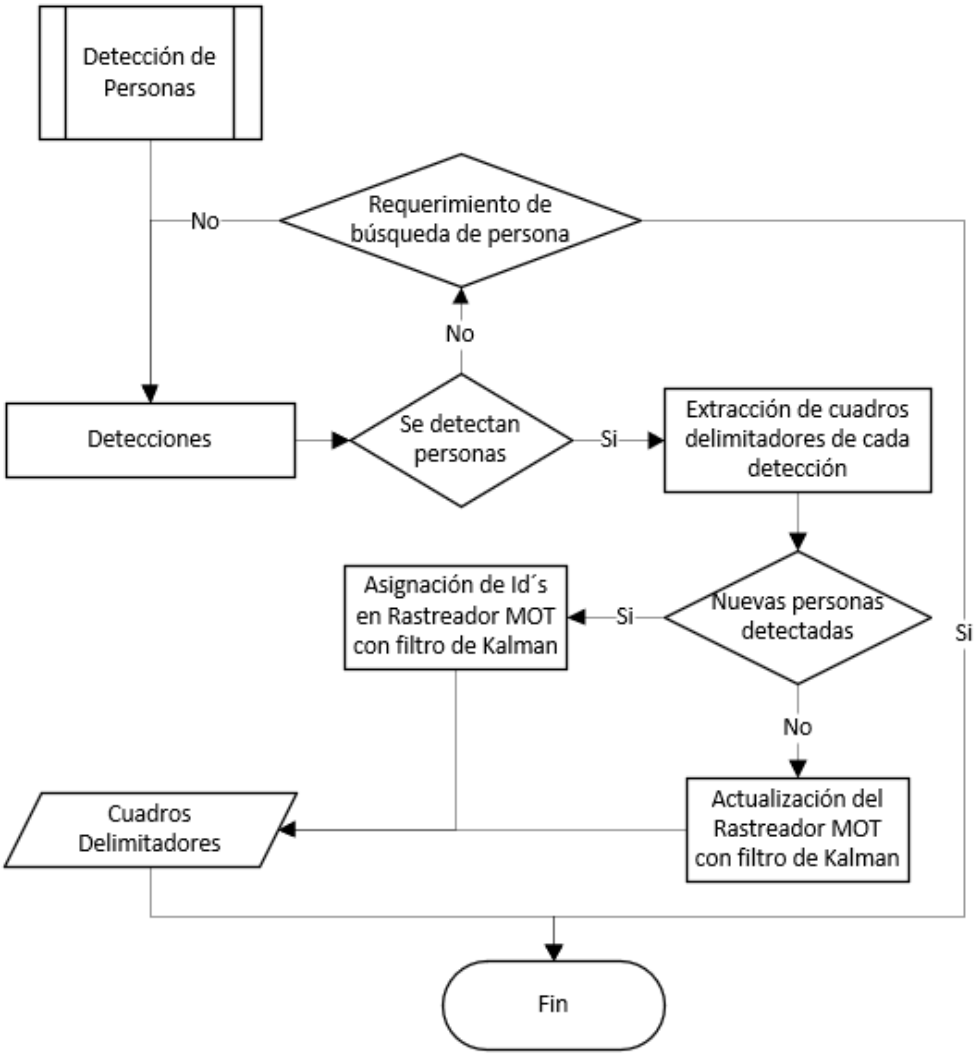
Seguimiento y rastreo de la persona seleccionada

Nota. El gráfico muestra la ejecución del algoritmo de seguimiento a una única persona, discriminando a las demás.

Los seguidores al igual que otros sistemas, tiene sus puntos débiles, uno de ellos es el problema de la oclusión o solapamiento del objeto, provocando la pérdida del Id asignado a este. La solución que permite mitigar en cierto grado este problema es un algoritmo de *Re-Identificación de persona* que se detalla en el Capítulo V.

Figura 38

Diagrama de flujo: detección y rastreo



Nota. El gráfico muestra el flujo del proceso de detección y rastreo de la persona a seguir.

Diseño del Controlador para el seguimiento con el micro-uav

En vista de la no linealidad del modelo de un UAV, se han desarrollado investigaciones con el fin de identificar el modelo dinámico lineal en base a la estimación de su no linealidad (Aguilar, Angulo, Costa, & Molina, 2014). Por otro lado, en (Aguilar, Salcedo, Sandoval, & Cobeña, 2017) (Salcedo, 2018) desarrollan un modelo basado en

video, que describe el movimiento de un UAV a partir del información visual recopilada de la cámara que lleva a bordo, obteniendo buenos resultados en comparación a un modelo hecho por mediciones inerciales que depende altamente de la fidelidad y tasa de transmisión de los datos, que en el caso del *micro-UAV* Bebop Parrot 2 está limitada a 5Hz. Por tales razones, este proyecto de investigación opta por realizar el modelado *SISO* del controlador del *micro-UAV* basado en video, como el desarrollado en (Salcedo, 2018).

En el seguimiento, las acciones de control son llevadas a cabo únicamente para la rotación alrededor del eje z, es decir el movimiento angular en "yaw" similar al seguimiento visual que realiza una persona al mover la cabeza en dirección al objetivo, para no perderlo de vista. La otra acción de control es el desplazamiento lineal en el eje "x", manteniendo una altura fija de vuelo de 1 metro.

Controlador del micro-uav en yaw

La metodología a usar para el modelado es similar a la de (Salcedo, 2018) pero con algunos ligeros cambios:

1. El sistema debe tener ya implementado las tareas de procesamiento de imágenes de la parte de detección, seguimiento, estimación de profundidad y búsqueda para poder obtener el tiempo de muestreo estimado de todo el sistema.
2. El método a usar para determinar el desplazamiento en pixeles de *frame* a *frame* ya no es el flujo óptico de Lucas Kanade o una transformación de similitud *Affine2D*, sino más bien se usa el algoritmo de detección como se observa en la Figura 39, situando a una persona en la parte central de la imagen y en función de las coordenadas (b_x, b_y) del centro del cuadro delimitador generado sobre

esta, se procede a tomar la variación de desplazamiento de sus centros, entre el $frame_t$ y el $frame_{t-1}$, mientras el *micro-UAV* realiza el movimiento angular en “z”.

Figura 39

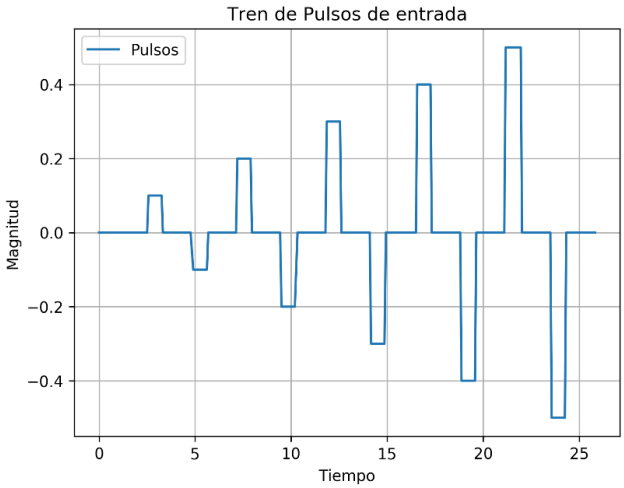
Obtención de la variación de desplazamientos entre frames en “yaw”

Nota. El gráfico muestra el proceso seguido para la obtención de los desplazamientos entre frames.

Una vez detectado y generado el cuadro delimitador, el *micro-UAV* es estimulado con un tren de impulsos como se ilustra en la Figura 40, abarcando el rango de velocidades permisibles que van desde [-1 ⋯ 1] para el parámetro *angular.z*. Debido a las configuraciones establecidas en la Tabla 4, el *micro-UAV* tiene una velocidad mucho mayor, por ende el rango máximo usado fue de 0.5, puesto que para valores mayores a este la persona quedaba fuera de la imagen.

Figura 40

Tren de pulsos para la estimación del modelo en “yaw”

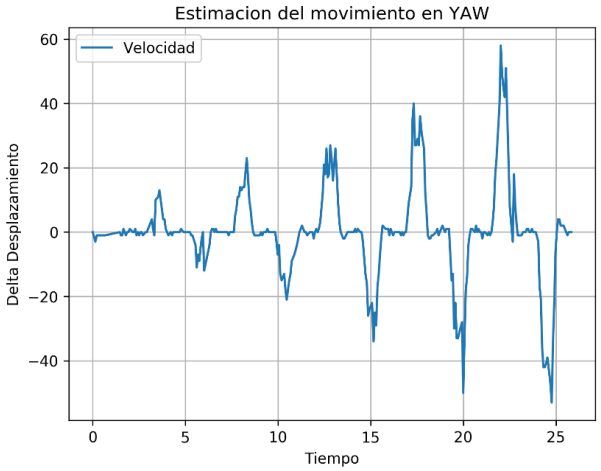


Nota. El gráfico muestra el tren de pulsos enviado al micro-UAV provocando movimientos de rotación en “yaw” durante 25 segundos.

El siguiente paso es el proceso de estimación del movimiento angular en “yaw”, para la identificación del modelo en este eje.

Figura 41

Estimación del movimiento en “yaw”



Nota. El gráfico muestra los resultados obtenidos del delta desplazamiento entre frames.

Lo siguiente a determinar el tiempo de muestreo a partir de la tasa de frames en 1 segundo (FPS) a la que se ejecuta todo el sistema, cuyo dato puede ser tomado de la Figura 39, mostrado en la parte superior izquierda de la imagen, que es de 20 FPS, cuyo tiempo de muestreo es calculado a través de la siguiente ecuación:

$$T_s = \frac{Tiempo\ transcurrido}{Numero\ de\ Frames\ Procesados} \tag{9}$$

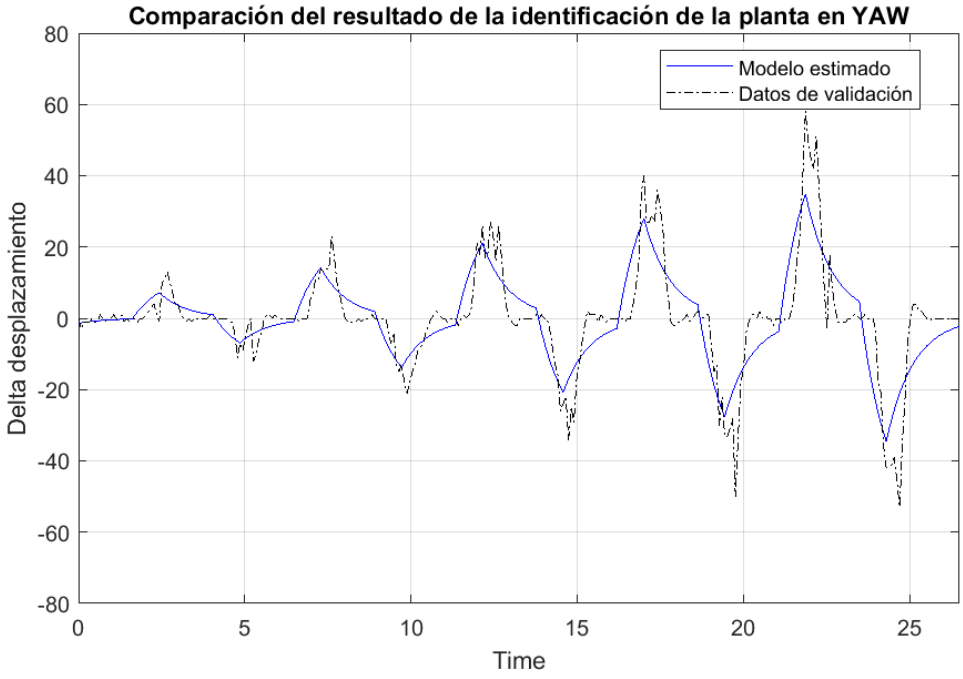
El tiempo de muestreo obtenido es de 50 ms. Esto quiere decir que el sistema tarda 50 ms en capturar y procesar cada frame. Con este dato y con la información de la Figura 40 y 41, junto con el uso del *Toolkit ident* de Matlab, se realiza la identificación de la planta. Que da lugar a una función de transferencia de primer orden que tiene la siguiente forma:

$$G_1(s) = \frac{K_p}{1 + T_p s} \tag{10}$$

Los valores correspondientes a cada parámetro son: $K_p = 115.3,\ T_p = 0.052$, de donde se obtiene la Figura 42 del modelo estimado.

Figura 42

Resultados de la identificación de la planta en "yaw"



Nota. El gráfico muestra los resultados obtenidos de la identificación de la planta a partir de los datos obtenidos durante 25 segundos de desplazamientos angulares en "yaw".

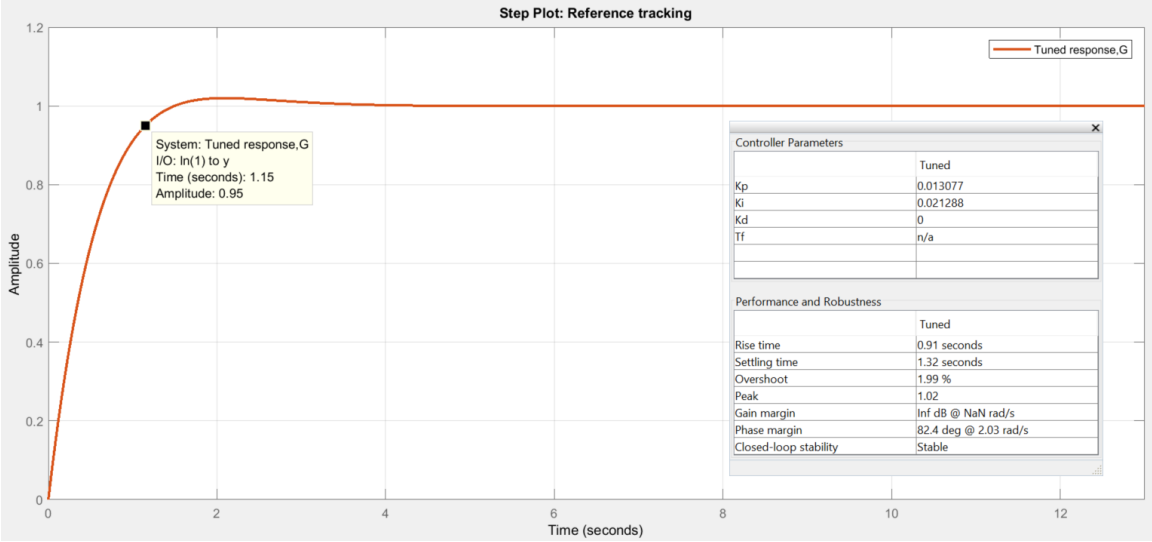
La siguiente etapa es el diseño del controlador que tenga una respuesta rápida y que se capaz de garantizar un error en estado estable cercano a cero. El diseño propuesto es un controlador PID, donde la parte derivativa se encargará de evitar posibles oscilaciones que se presenten en el sistema. Este contralor es sintonizado a través de la herramienta de Matlab *pidTuner*, con las siguientes consideraciones de rendimiento:

$$M_p = 2\%\,, \quad ts \leq 2\ seg \tag{11}$$

Los resultados de las ganancias obtenidas de la sintonización del controlador PID son las siguientes: $K_p = 0.013077,\ K_i = 0.021288$ y $K_d = 0.0$.

Figura 43

Respuesta al escalón unitario del sistema Planta-Controlador en "yaw"

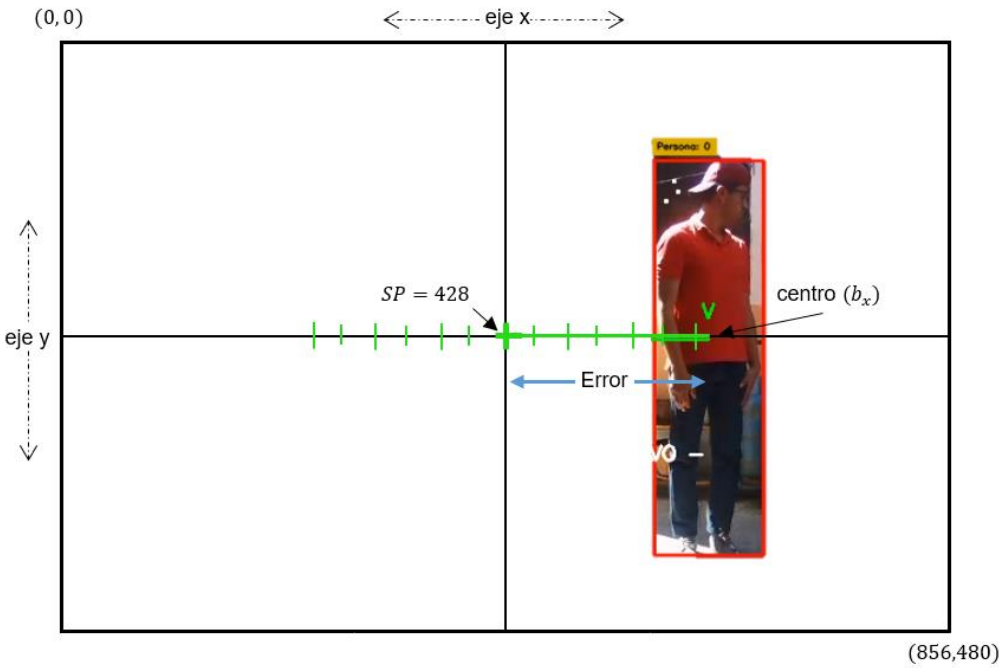


Nota. El gráfico muestra la respuesta ante una entrada escalón unitario durante 12 segundos.

Para realizar la acción de control el setpoint (SP) se define como el pixel central de toda la imagen que es el valor de $428px$. Con ello el calculo de error de posición se determina a partir de la variación en pixeles que existe entre el setpoint y las coordenadas (b_x, b_y) del centro del cuadro delimitador generado sobre la persona a seguir como se muestra en la Figura 44. Con estos datos el controlador procesa y envía instrucciones de control al *micro-UAV* para que gire ya sea a la izquierda o derecha, con el fin de que la persona se encuentre centrada en la imagen.

Figura 44

Determinación del error para la acción de control en "yaw"

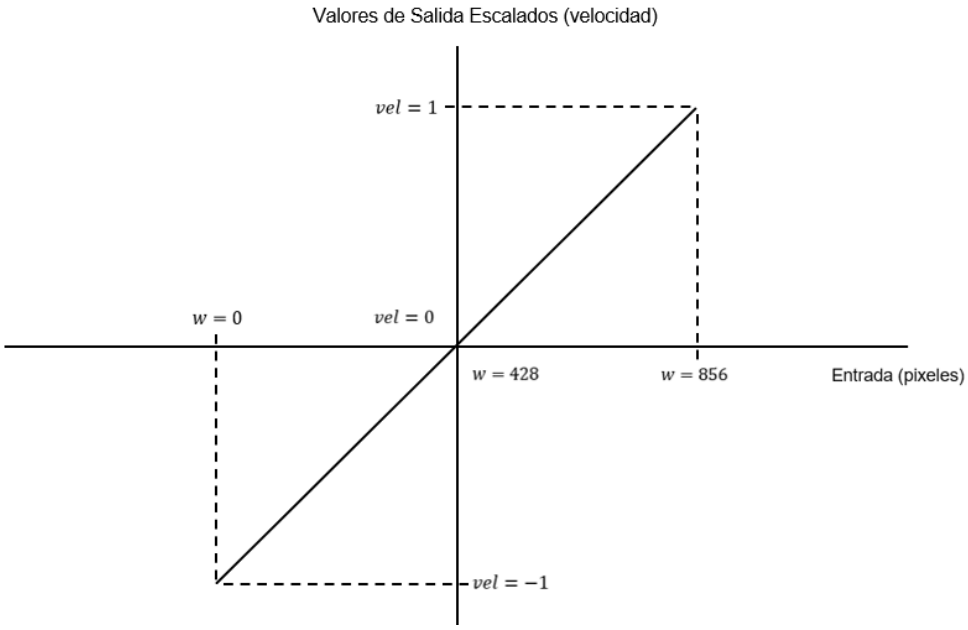


Nota. El gráfico muestra el proceso para determinar la distancia de separación en pixeles desde el centro de la imagen al centro del cuadro delimitador.

Los valores en pixeles recibidos de las imágenes procesadas que ingresan al controlador son traducidos y escalados a valores manejables de velocidades del *micro-UAV* en el intervalo de [-1 ⋯ 1]. En la Figura 45 se presenta el *Escalado Bipolar* (escalado en el rango de valores positivos y negativos) responsable de este proceso. De tal forma que aseguramos que le controlador envíe valores dentro del rango aceptable de velocidades del *micro-UAV*

Figura 45

Escalado Bipolar de pixeles a velocidades admitidas en "yaw"



Nota. El gráfico muestra la recta que rige el proceso de escalado de pixeles a valores de velocidad permitidas por el micro-UAV Parrot Bebop 2.

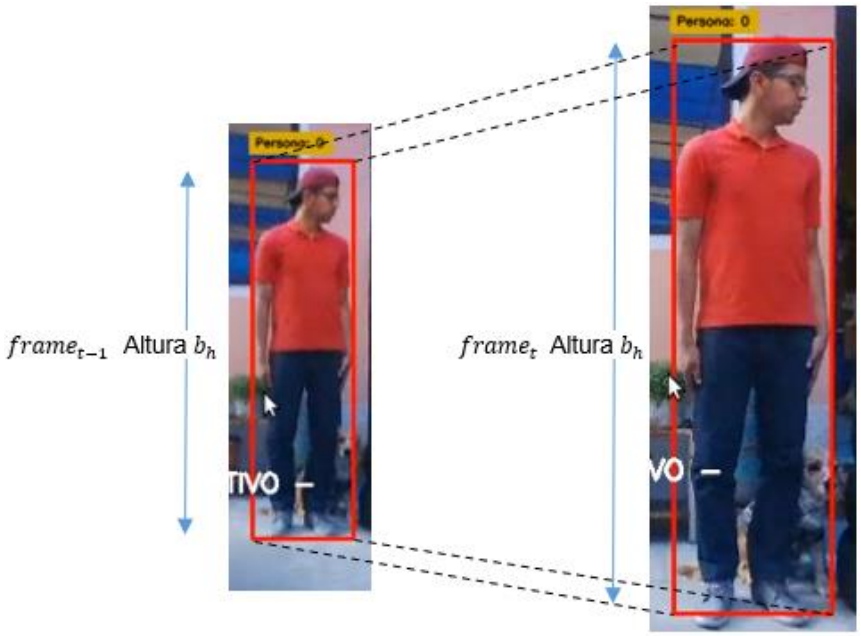
Controlador del micro-uav en pitch

La metodología a usar mantiene el primer punto del controlador anterior, el cambio se da en el segundo punto por lo siguiente:

- A partir de la detección de la persona como se observa en la Figura 46, se toma la variable b_h que representa el alto del cuadro delimitador generado sobre esta, se procede a tomar la variación de desplazamiento de esta altura entre el $frame_t$ y el $frame_{t-1}$, mientras el *micro-UAV* realiza el movimiento de desplazamiento lineal en "z". De esta forma tendremos un estimado de si la persona se aleja o se acerca.

Figura 46

Obtención de la variación de desplazamientos entre frames en “pitch”



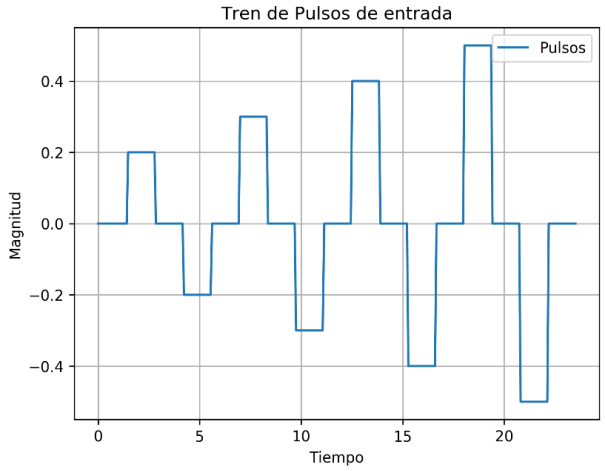
Nota. El gráfico muestra la variación de altura del cuadro delimitador al alejarse y acercarse el micro-UAV a la persona.

Otra solución para esta misma tarea pudo ser en base al área del cuadro delimitador, pero esta opción no es muy viable debido a que, si la persona extiende sus brazos horizontalmente, el cuadro delimitador se vuelve más ancho y estaría generando un área mucho mayor, lo que llevaría a confundir al algoritmo.

Una vez detectado y generado el cuadro delimitador, el *micro-UAV* es estimulado con un tren de impulsos como se ilustra en la Figura 46, abarcando el rango de velocidades permisibles que van desde [-0.5 ⋯ 0.5] para el parámetro *linear.x*. Debido a las configuraciones establecidas en la Tabla 4, el *micro-UAV* tiene una velocidad mucho mayor, por ende el rango máximo usado fue de 0.5, puesto que para valores mayores a este, el vuelo se torna muy brusco y a la vez peligroso ante alguna posible colisión.

Figura 47

Tren de pulsos para la estimación del modelo en “pitch”

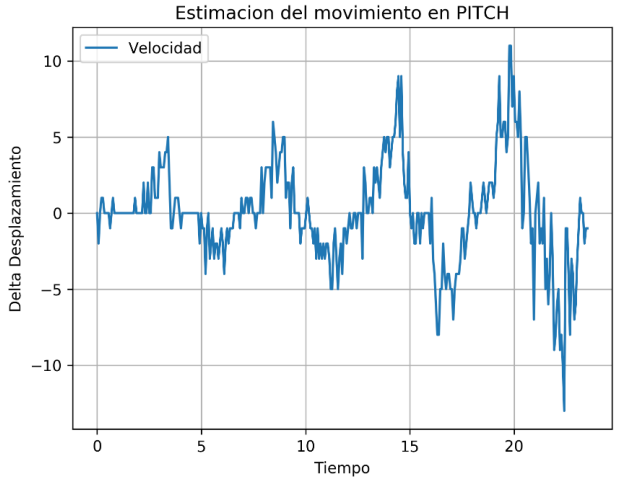


Nota. El gráfico muestra el tren de pulsos enviado al micro-UAV provocando movimientos de desplazamientos en el eje “x” durante 25 segundos.

El siguiente paso es el proceso de estimación del movimiento lineal en el eje “x”, para la identificación del modelo en este eje.

Figura 48

Estimación del movimiento en “pitch”



Nota. El gráfico muestra los resultados obtenidos del delta desplazamiento entre frames.

Al igual que el controlador en “yaw” la tasa obtención de las imágenes es de 20 FPS, cuyo cálculo del tiempo de muestro sigue siendo de 50 ms.

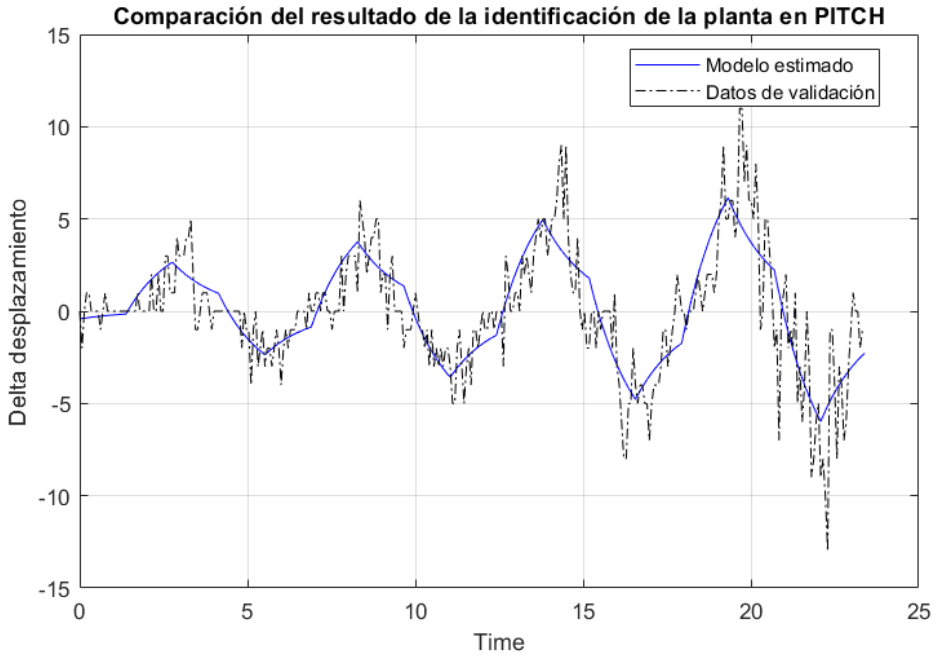
Con la información de la Figura 47 y 48, junto con el uso del *Toolkit* de Matlab, se realiza la identificación de la planta. Que da lugar a una función de transferencia de primer orden que tiene la siguiente forma:

$$G_2(s) = \frac{K_p}{1 + T_p s} \tag{12}$$

Los valores correspondientes a cada parámetro son: $K_p = 21.28,\ T_p = 0.051$, de donde se obtiene la Figura 49 del modelo estimado.

Figura 49

Resultados de la identificación de la planta en “pitch”



Nota. El gráfico muestra los resultados obtenidos de la identificación de la planta a partir de los datos obtenidos durante 25 segundos de desplazamientos lineales en el eje “x”.

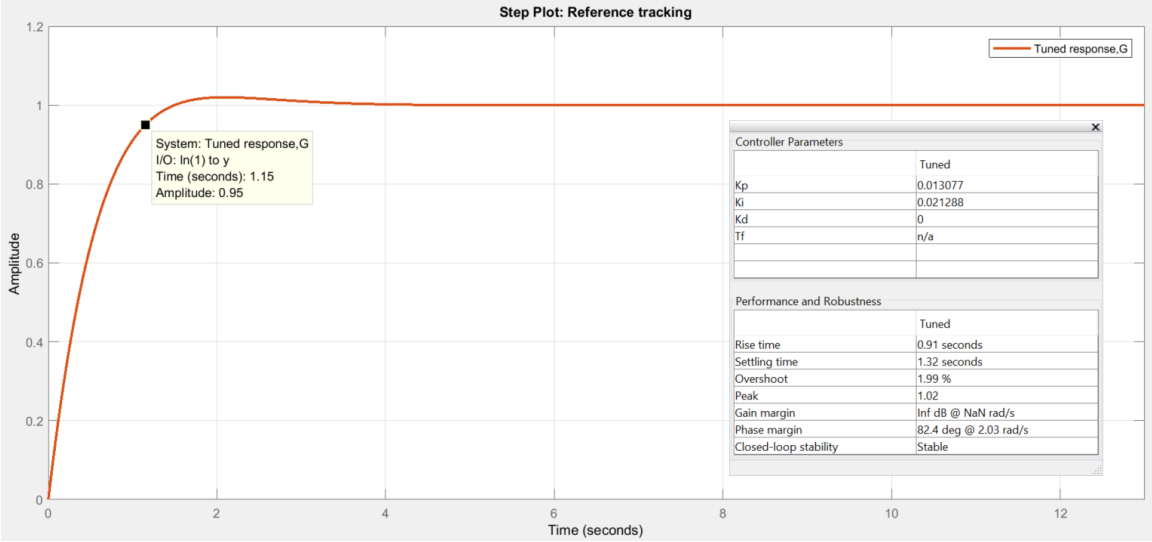
La siguiente etapa es el diseño del controlador que tenga una respuesta rápida y que se capaz de garantizar un error en estado estable cercano a cero. El diseño propuesto es un controlador PID, donde la parte derivativa se encargará de evitar posibles oscilaciones que se presenten en el sistema. Este contralor es sintonizado a través de la herramienta de Matlab *pidTuner*, con las siguientes consideraciones de rendimiento:

$$M_p = 2\%\,, \quad ts \leq 2\ seg \tag{13}$$

Los resultados de las ganancias obtenidos de la sintonización del controlador PID son las siguientes: $K_p = 0.12213$, $K_i = 0.079177$ y $K_d = 0.0$. Dando una respuesta ante una entrada impulso como la que se muestra a continuación.

Figura 50

Respuesta al escalón unitario del sistema Planta-Controlador en "pitch"

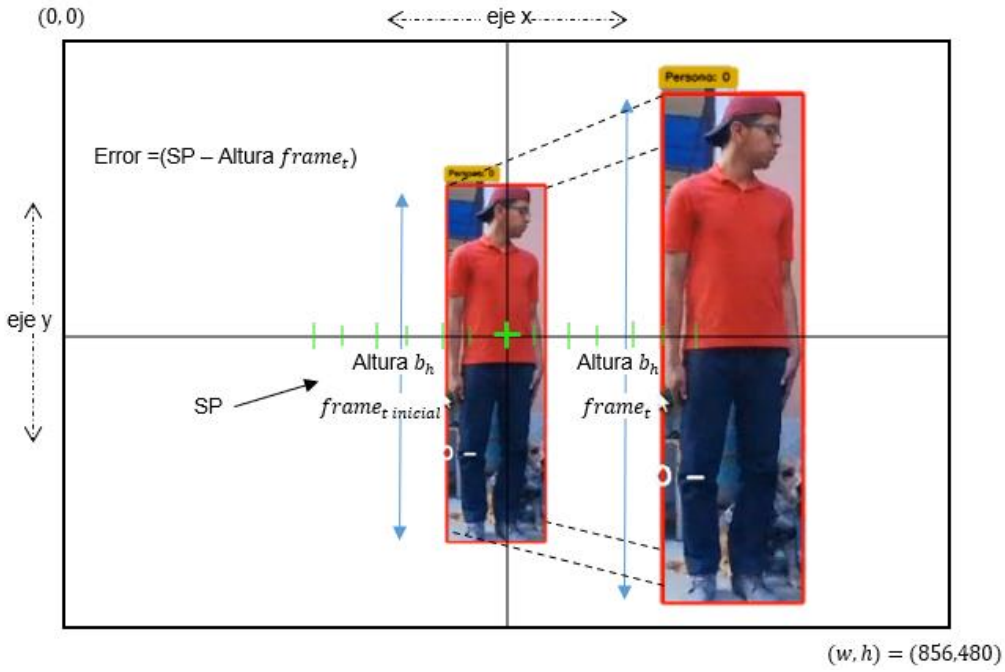


Nota. El gráfico muestra la respuesta ante una entrada escalón unitario durante 12 segundos.

Para realizar la acción de control, el setpoint se define como el valor en pixeles de la altura b_h del cuadro delimitador en el frame inicial, este valor no es fijo, más bien depende de la distancia a la que se encuentre la persona del *micro-UAV*, por lo que tratara de seguirlo a esa distancia de separación. Con ello el cálculo de error de la altura se determina a partir de la variación en pixeles que existe entre el setpoint y las alturas del cuadro delimitador en los frames posteriores como muestra en la Figura 51. Con estos datos el controlador procesa y envía instrucciones de control al *micro-UAV* para que se acerque o se aleje de la persona, con el fin de mantener el seguimiento a una distancia adecuada.

Figura 51

Determinación del error para la acción de control en "pitch"

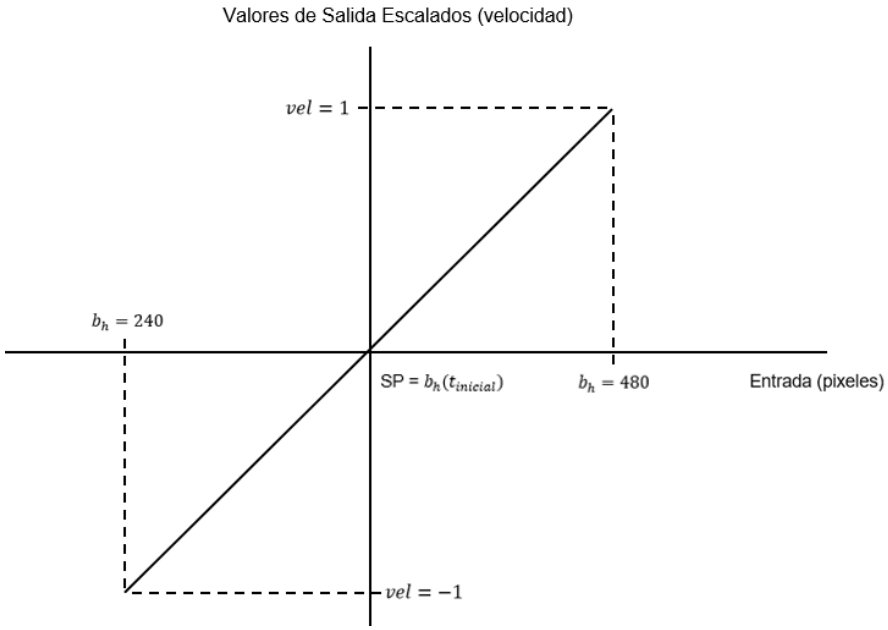


Nota. El gráfico muestra el proceso para determinar la distancia de separación en pixeles de la altura entre el cuadro delimitador del frame actual y el anterior.

De la misma manera que en el controlador anterior en “yaw”, los valores en pixeles recibidos de las imágenes procesadas que ingresan al controlador son traducidos y escalados a valores manejables de velocidades del *micro-UAV* en el intervalo de [-1 ⋯ 1] mediante el *Escalado Bipolar* como se ilustrada en la Figura 52. Se considera que la persona no se va encontrar a más 5 metros de distancia del *micro-UAV*, por ende, la altura del cuadro delimitador no va ser menor a la mitad del alto de la imagen, quedando así el intervalo de alturas entre $240px$ y $480px$.

Figura 52

Escalado Bipolar de pixeles a velocidades admitidas en “pitch”

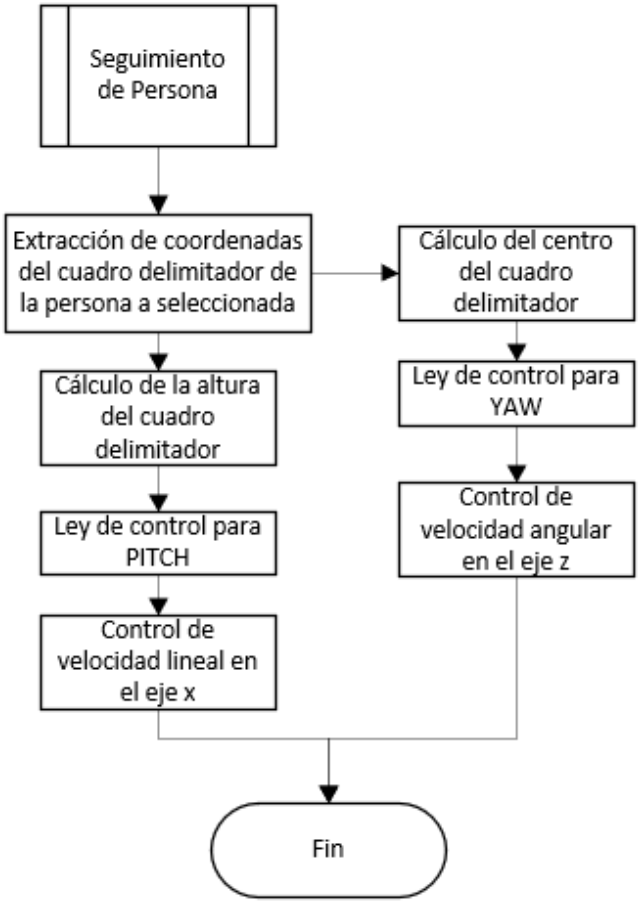


Nota. El gráfico muestra la recta que rige el proceso de escalado de pixeles a valores de velocidad permitidas por el micro-UAV Parrot Bebop 2.

Finalmente, en la Figura 53 se muestra el proceso de seguimiento de persona junto con sus controladores en “yaw” y “pitch”, esto detallado mediante un diagrama de flujo.

Figura 53

Diagrama de flujo: seguimiento de persona



Nota. El grafico corresponde a un diagrama de flujo de la secuencia que debe seguir el proceso de seguimiento de una persona con movimientos en “pitch” y “yaw”

Evasión de obstáculos

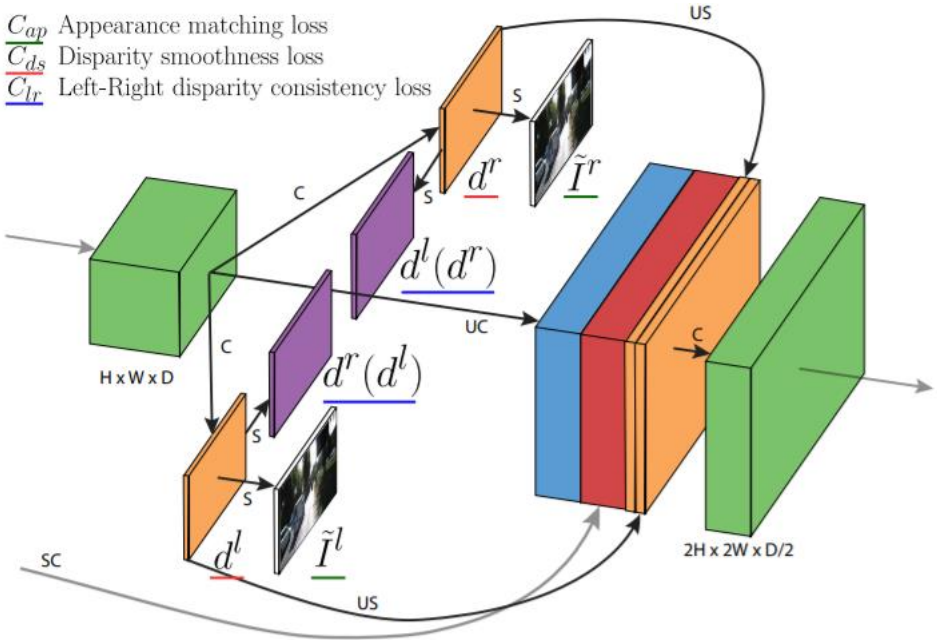
Una de las consideraciones más importantes en un vehículo autónomo es que cuente con la habilidad de reaccionar ante posibles objetos que obstruyan el camino durante el seguimiento, es decir que pueda evadir obstáculos presentes en su entorno para así evitar lastimar alguna persona, evitar colisiones y daños en la estructura del vehículo (Aguilar & Morales, 3D Environment Mapping Using the Kinect V2 and Path

Planning Based on RRT Algorithms, 2016),(Cabras, Rosell, Pérez, Aguilar, & Rosell, 2011),(Aguilar, Abad, Ruiz, Aguilar, & Aguilar-Castillo, 2017),(Aguilar, Morales, Ruiz, & Abad, 2017),(Aguilar, Sandoval, Limaico, Villegas-Pico, & Asimbaya, 2019).

Esto es posible gracias a algoritmos de *estimación de profundidad*, basados en *Deep Learning* y una arquitectura *Fully Convolutional Network* como se especifica en la fundamentación teórica. El algoritmo toma una imagen RGB 2D y genera a partir de ello una imagen de profundidad en escala de grises con información estimada sobre la distancia a la que se encuentran los objetos.

Figura 54

Módulo de pérdidas que genera mapas de disparidad"



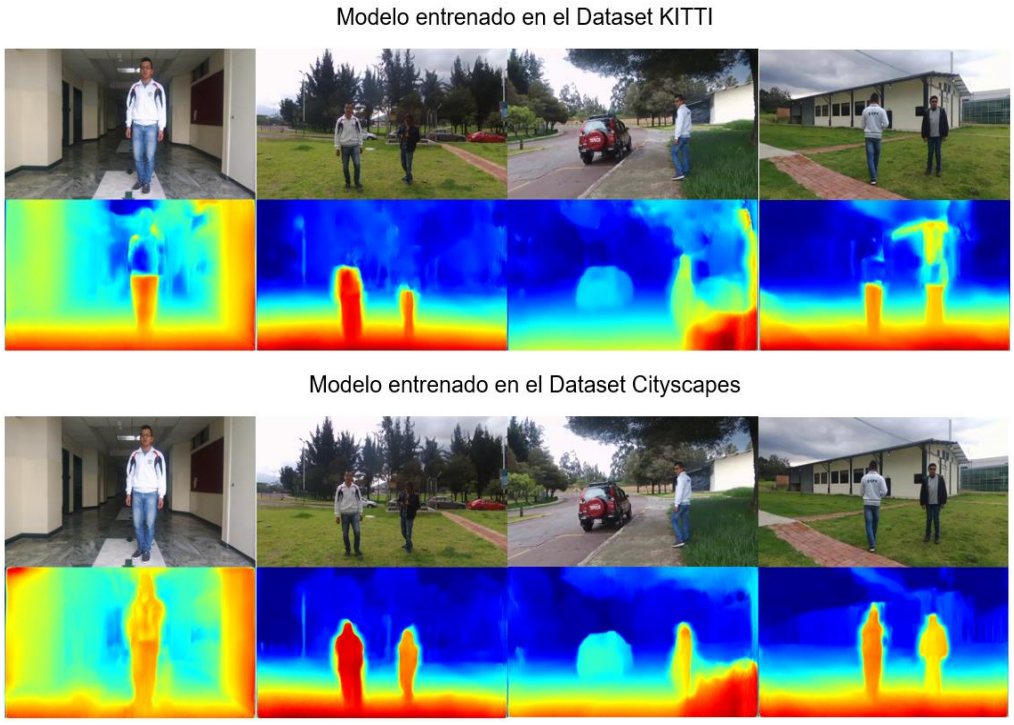
Nota. El gráfico muestra cómo se forma el mapa de disparidad a partir de dos imágenes estéreo. Tomado de *Unsupervised Monocular Depth Estimation with Left-Right Consistency,* (Godard, Aodha, & Brostow, 2017).

El modelo de la Figura 54 es el encargado de esta tarea, mismo que consta de una *red neuronal convolucional* conocida como monodepth, entrenada con imágenes de pares estéreo rectificados. En sí se trata de una estimación de profundidad monocular no supervisada con consistencia izquierda-derecha (Godard, Aodha, & Brostow, 2017)

El proceso fundamentalmente consiste en usar imágenes estéreo, una izquierda y una derecha, e intentar predecir y reconstruir la otra a partir de una de ellas, generando un mapa de disparidad mediante un módulo de pérdidas. Esta información junto con la distancia focal de la cámara y la distancia entre las cámaras estéreo usadas en el entrenamiento, es aprovechada para estimar la profundidad.

Figura 55

Estimación de profundidad con el modelo monodepth



Nota. El gráfico muestra los resultados de las inferencias obtenidas al ejecutar el modelo de estimación de profundidad con dos distintos tipos de datasets.

El estado del arte de modelos de estimación de profundidad es muy amplio, no obstante, la elección de este modelo es en virtud de la compatibilidad que ofrece con los frameworks y softwares con los que cuenta la *estación en tierra*, además de los resultados muy prometedores que muestra. La implementación del modelo es hecha a través de *TensorFlow*, disponible para los dataset KITTI y Cityscapes, identificando una mejor estimación con Cityscapes.

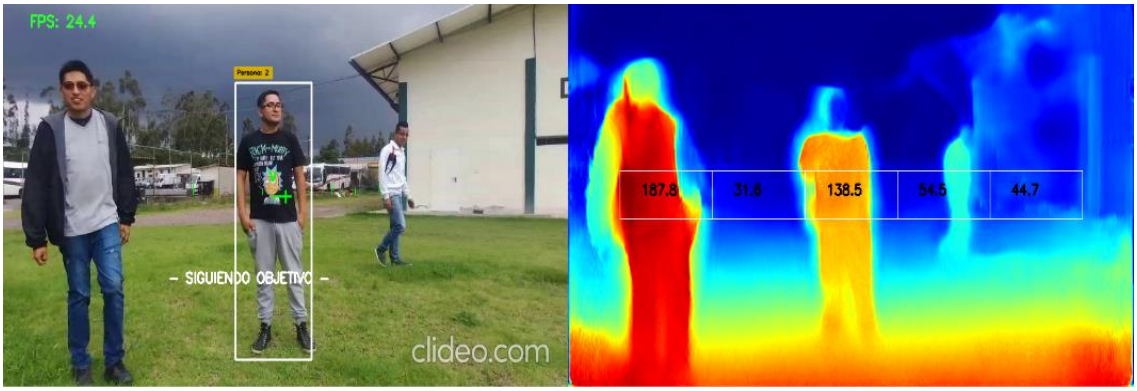
Las secciones de la Figura 55 muestra la estimación de profundidad en una imagen, donde los pixeles con tintes rojizos representan los objetos más cercanos, mientras que los más alejados tienden a estar representados por pixeles más oscuros, como el azul. Todo este proceso es ejecutado en paralelo mediante *Hilos* con el fin de no congestionar los procesos de detección y seguimiento.

Lógica de evasión de obstáculos

En base a la información de la variación del color de los pixeles que genera el modelo, se establece un rango de variación mínima para considerar que zona es libre de colisión, caso contrario para zonas con variaciones mayores a este rango se consideran como un obstáculo presente. Para ello como se ilustra en la Figura 56 se analizan cinco tramos a lo ancho de la imagen (rectángulos), pero únicamente de la parte central de esta, porque representa las zonas por las cuales el *micro-UAV* circulará en vista que el desplazamiento lo realiza a una altura fija de 1 metro. En consecuencia, los demás tramos de la imagen tanto superior e inferior no se toman en cuenta en el análisis puesto que no representan un obstáculo para el *micro-UAV*.

Figura 56

Análisis de una zona libre de colisión tramo a tramo en la imagen

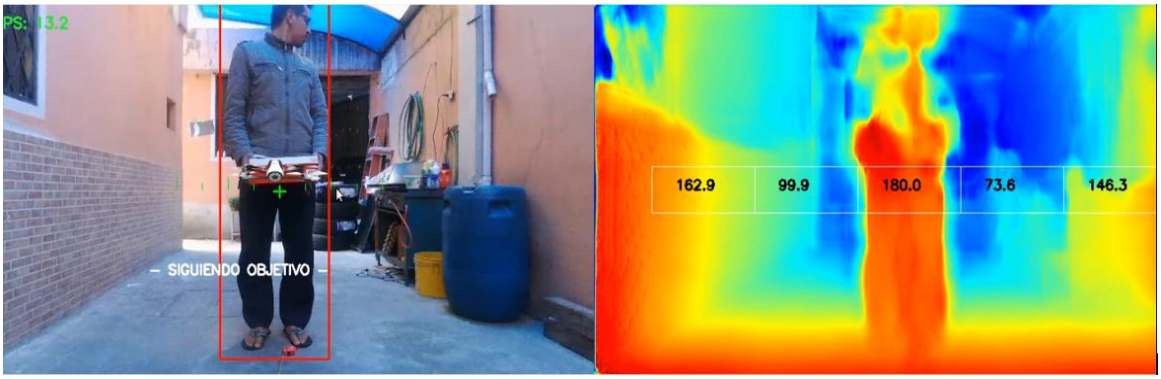


Nota. El gráfico muestra los tramos en los cuales se determina los valores de profundidad.

La determinación del ancho y alto de cada rectángulo que representa un tramo a analizarse viene en función del espacio ocupado en pixeles por el *micro-UAV* a una distancia de 2m, que prácticamente es similar al espacio ocupado por una persona a esa misma distancia. Los resultados se observan en la Figura 57, en la que se utiliza una caja con las dimensiones similares a las del *micro-UAV* tanto en alto como ancho para simular la presencia del mismo. Quedando delimitado de dimensiones de 60 pixeles de alto y 140 pixeles de ancho para cada rectángulo.

Figura 57

Determinación de los tramos que analizan profundidad



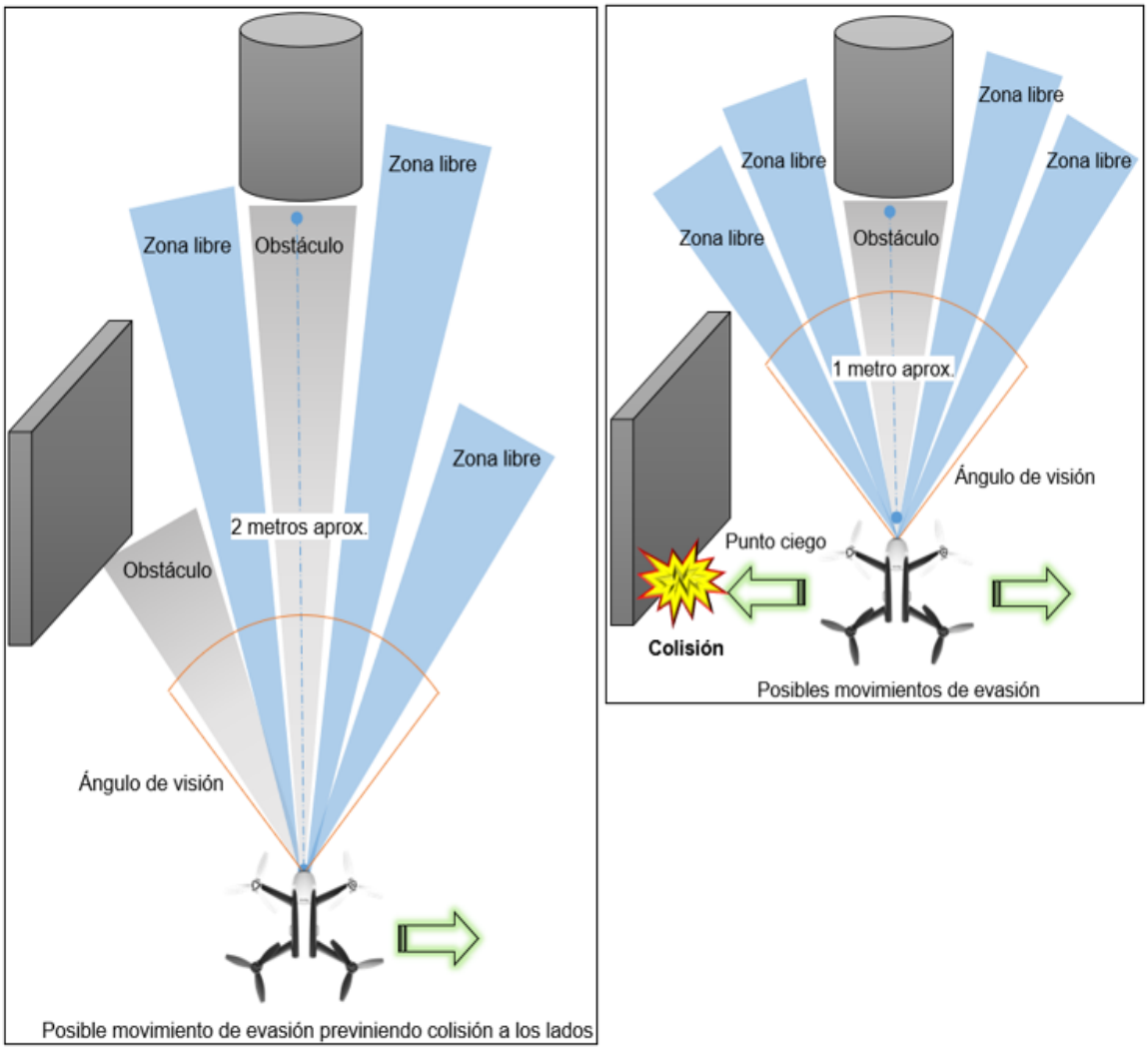
Nota. El gráfico justifica las dimensiones de los rectángulos que delimitan cada tramo.

El *micro-UAV* responde a una evasión de obstáculos y corrección de su trayectoria a una distancia aproximada de 2m, establecida como una distancia segura y adecuada puesto que tiene un panorama mucho más amplio del entorno y de los obstáculos que se encuentran presentes en el rango de visión de la cámara que es de 80° horizontal. Mientras que si se reduce esta distancia el *micro-UAV* se acercará más al obstáculo antes de responder a la orden de evasión, pero esto ocasionará el perder de vista a los posibles obstáculos que se encuentran a los extremos que pueden ocasionar una colisión y que pudieron ser previsto con una respuesta temprana, con una evasión a mayor distancia del obstáculo.

En la Figura 58 se ilustra este comportamiento desde una vista superior, en donde se indican zonas libres representadas en color celeste y los posibles movimientos de evasión ante la detección de obstáculos representados por color plateado.

Figura 58

Acción de evadir obstáculos a 2 metros y 1 metro.



Nota. El gráfico justifica el proceso de evadir obstáculos a una distancia de aproximadamente 2 metros, por precaución.

El cálculo de si un obstáculo está a una distancia aproximada de 2 metros o menor a esta, es llevado a cabo en función del promedio de la intensidad de los pixeles que se encuentran en cada tramo de análisis de la Figura 57 y 58 mediante el algoritmo de la Figura 59. El rango de variación de cada pixel va desde 0 a 255 debido a que la imagen

es transformada a escala de grises para su análisis en un solo canal de color, es decir la variación ocurre desde un pixel blanco hasta un pixel completamente negro.

Figura 59

Algoritmo para el cálculo de las intensidades en cada rectángulo

Algorithm 1: Promedio de intensidades en cada rectángulo

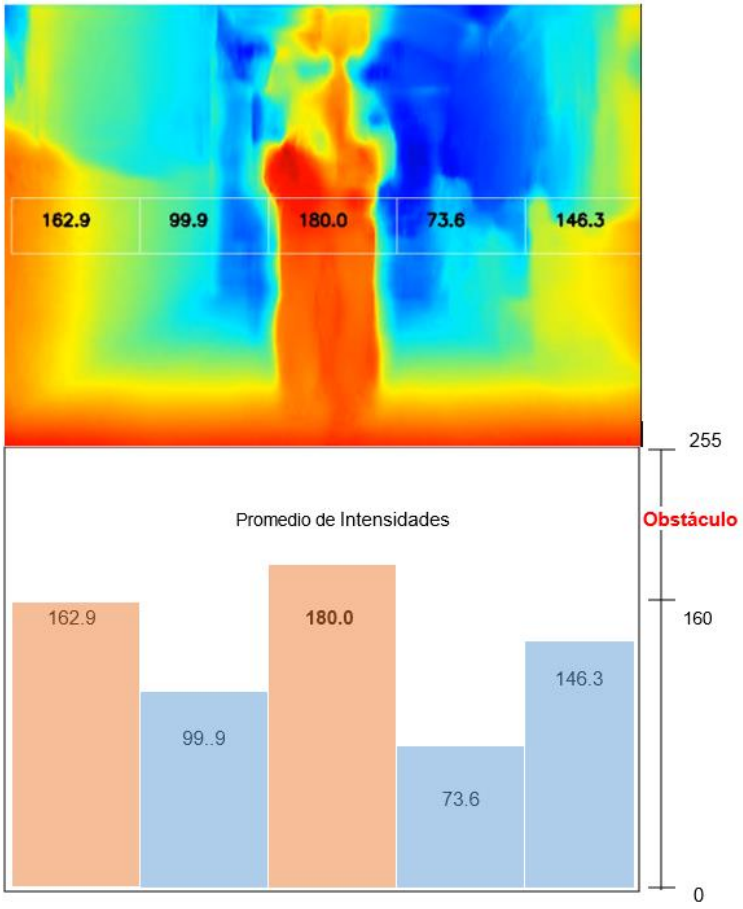
```
Input = imagen.profundidad;
inicialización;
pixel.ancho.ini = 78;
pixel.ancho.fin = 218;
pixel.altura.ini = 210;
pixel.altura.fin = 270;
ancho = 140;
alto = 60;
sum.pixel = 0;
prom.pixel = [ ];
for num.rectangulo = 0 to 5 do
    for i = pixel.ancho.ini to pixel.ancho.fin do
        for j = pixel.altura.ini to pixel.altura.fin do
            pixel = imagen.profundidad[j,i];
            sum.pixel = sum.pixel + pixel;
        end
    end
    prom.pixel[num.rectangulo] = sum.pixel/(ancho*alto);
    pixel.ancho.ini = pixel.ancho.fin;
    pixel.ancho.fin = pixel.ancho.fin + ancho;
end
return promedio.pixel
```

Nota. En el gráfico se presenta una descripción en alto nivel o también llamado pseudocódigo, del proceso de cálculo de las intensidades.

Los resultados de cada tramo son interpretados como un mapa de profundidad que indica la zona libre de colisión, representada por niveles de intensidades como se muestran en la Figura 60. Esta información obtenida corresponde a un obstáculo que se encuentra a 2m del *micro-UAV*

Figura 60

Mapa de profundidad y niveles de intensidades de un obstáculo a 2m



Nota. El gráfico corresponde a un mapa de profundidad o disparidad, junto con un gráfico de barras, que simboliza las zonas con obstáculos (barras naranjas) y zonas libres (barras azules).

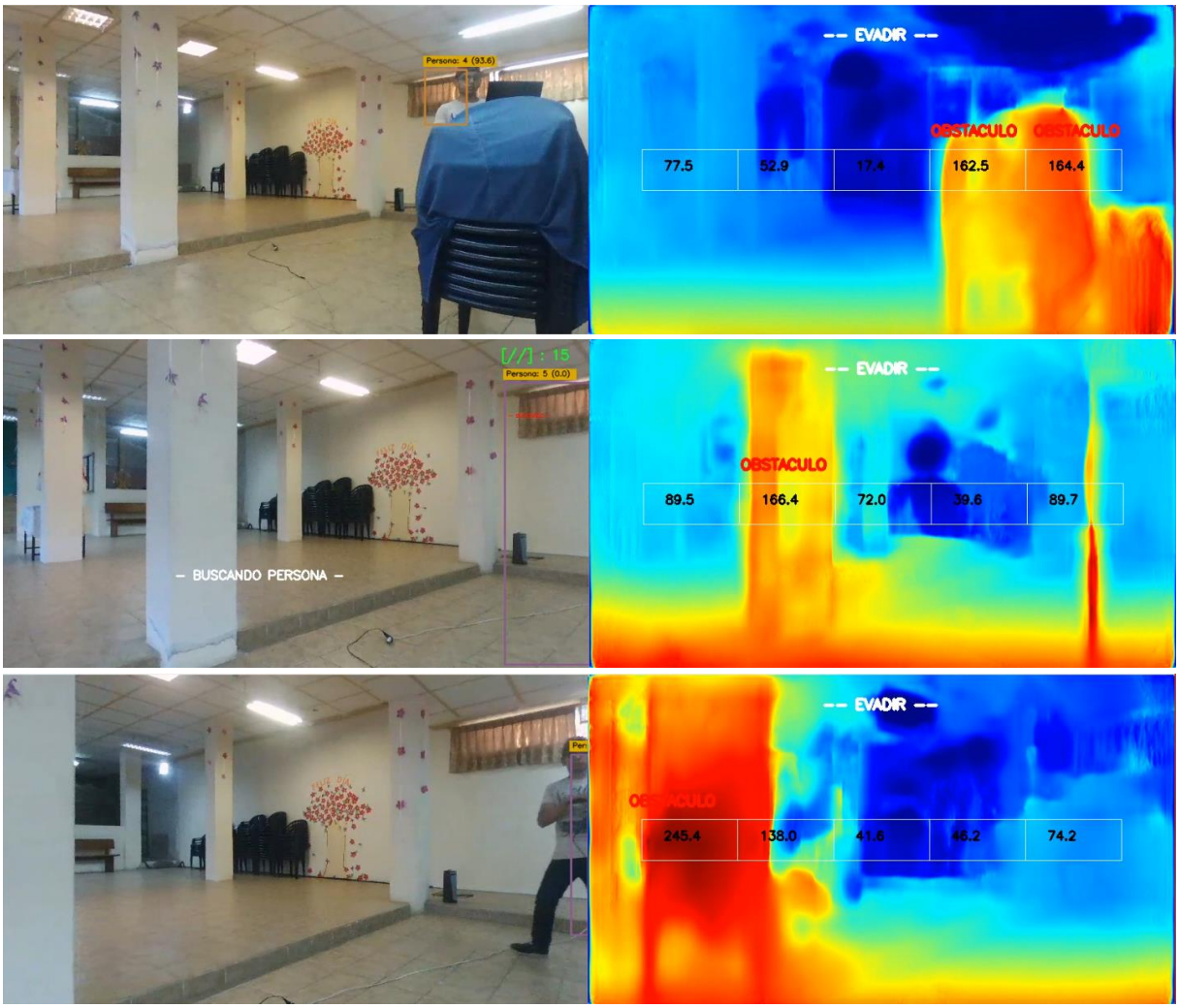
De los resultados anteriores se establece el valor de 160 como un umbral decisión, en el que valores mayores a este son considerados obstáculos y por lo tanto se ejecuta la acción de evadir.

Acción de control para la evasión de obstáculos

La acción de evasión es llevada a cabo mediante un controlador ON-OFF que actúa únicamente al detectar un obstáculo como se observa en la Figura 61, enviando un valor fijo de 0.1 o -0.1 al parámetro *linear.y* que genera un desplazamiento horizontal en dirección contraria al obstáculo.

Figura 61

Resultados de la estimación de profundidad y evasión de obstáculos

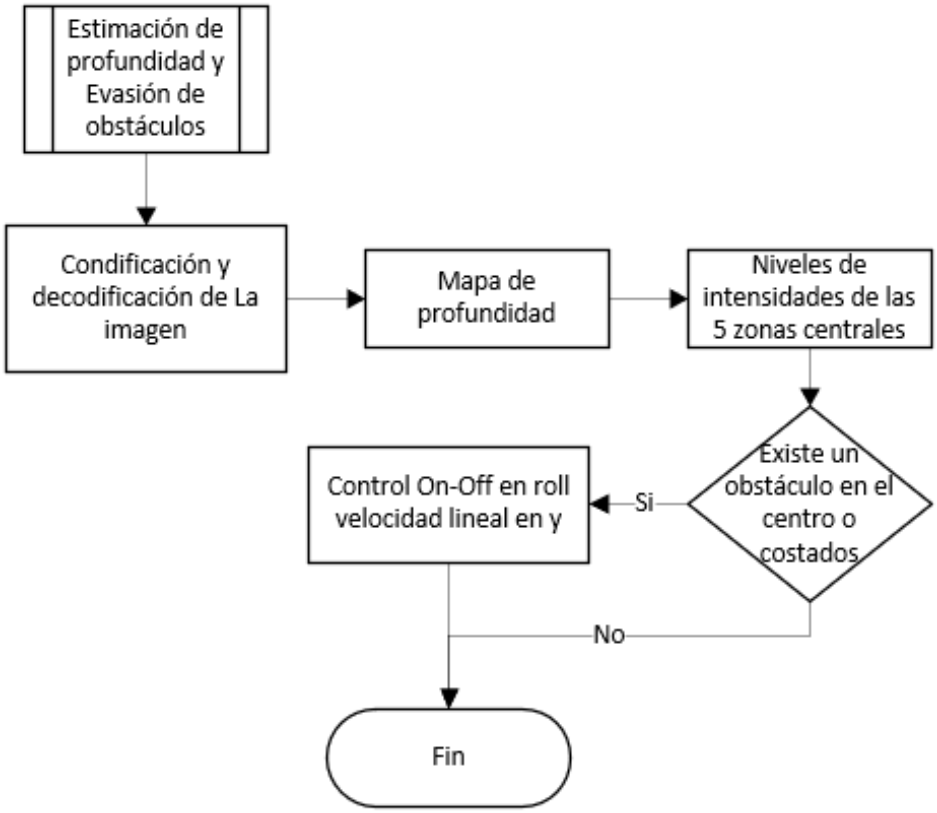


Nota. En el gráfico muestra las alertas visuales que se generan cuando se han detectado obstáculos, lo que va acompañado de una acción de evasión.

En el caso de perder de vista a la persona, el *micro-UAV* realiza intervalos de desplazamientos en la dirección donde se obtuvo la última detección, después de cada intervalo el *micro-UAV* ejecuta un pequeño giro en “yaw” de forma horaria o antihoraria según la dirección de su ultimo rastro, para así intentar encontrarla.

Figura 62

Diagrama de flujo: estimación de profundidad y evasión de obstáculos



Nota. El gráfico corresponde al diagrama de flujo de la secuencia que debe seguir el proceso de estimación de profundidad para la evasión de obstáculos.

Capítulo V

Búsqueda de Persona

Durante el seguimiento que realiza el *micro-UAV* a la persona objetivo pueden ocurrir distintos eventos que interrumpan su visibilidad, tales como oclusiones, cambios en la iluminación, giros o movimientos bruscos y demasiado rápidos. Para afrontar esta situación es necesario implementar una estrategia de búsqueda del objetivo a partir de sus características visuales, como el color, forma, líneas, texturas entre otros patrones que permitan identificarlo. Como bien se conoce esta tarea puede ser abordada por uno de los modelos más potentes en el reconocimiento de patrones en imágenes conocida como *red neuronal convolucional* y así partir con la creación y entrenamiento de un clasificador de imágenes que permita obtener una estimación de si la imagen corresponde o no a la persona buscada. Esta estrategia es conocida en el campo de *computer vision* como *Re-Identificación de persona*.

Re-Identificación de Persona

La estrategia se fundamenta en recolectar imágenes de distintos ángulos de una persona para posteriormente asociarlas con nuevas capturas e intentar identificarla. En consecuencia, se convierte en un problema de clasificación, por lo que precisa del entrenamiento de un modelo para la clasificación de las imágenes, partiendo de la base teórica referente a la estructura de una modelo *red neuronal convolucional*. El proceso se detalla a continuación.

Recolección de Imágenes

El proceso de obtención de las imágenes correspondientes a la persona y la creación de un Dataset, entra en ejecución una vez hecha la detección y la selección para

su seguimiento. En esta instancia el sistema captura 50 imágenes del objetivo a seguir durante 10 segundos, intervalo de tiempo necesario para que la persona gire 360° sobre su eje, posibilitando así capturas de sus diferentes lados.

Figura 63

Recolección de imágenes de la persona a seguir

Nota. En el gráfico se presenta un ejemplo de las muestras tomadas en una de las pruebas.

Creación del modelo de clasificación de imágenes

La implementación del modelo es realizada mediante el uso de los frameworks *TensorFlow* y *Keras* como backend del sistema, a partir de la guía de (TensorFlow.org, 2019). Cabe mencionar que no hay una configuración fija para clasificadores de imágenes, tanto los parámetros, hiperparámetros y las configuraciones normalmente se van ajustando a medida que se reentrena y se evalúa el modelo, hasta obtener un rendimiento aceptable en la red, en cuanto a precisión. A continuación, se despliegan los pasos seguidos:

1. Inicialización de parámetros de entrenamiento: número de *épocas=20*, uno de los parámetros más importantes el *batch_size=16*, un tamaño bastante pequeño que posibilita una convergencia más rápida, a diferencia de un tamaño elevado que tiende a conducir a una generalización deficiente y demorosa.
2. El siguiente paso da lugar a la preparación de los datos en donde se lleva a cabo la práctica de *Data Augmentation*, implementado en la función "ImageDataGenerator()" de la Figura 64. Consiste en una forma efectiva de aumentar la cantidad de datos para el entrenamiento, muy útil al contar con una cantidad limitada de ellos, como en este caso.

   Esta técnica se encargará de generar réplicas del dataset de imágenes de entrenamiento, para posteriormente someterlas a variaciones y distorsiones como se observa en la Figura 65. Entre las usadas están: re-escalado, rotación, desplazamiento, recorte, zoom, volteos, rellenos y variación de iluminación. A esto se añade un proceso de normalización de $1/255$ para el Dataset de validación.

Figura 64

Función Data Augmentation y normalización

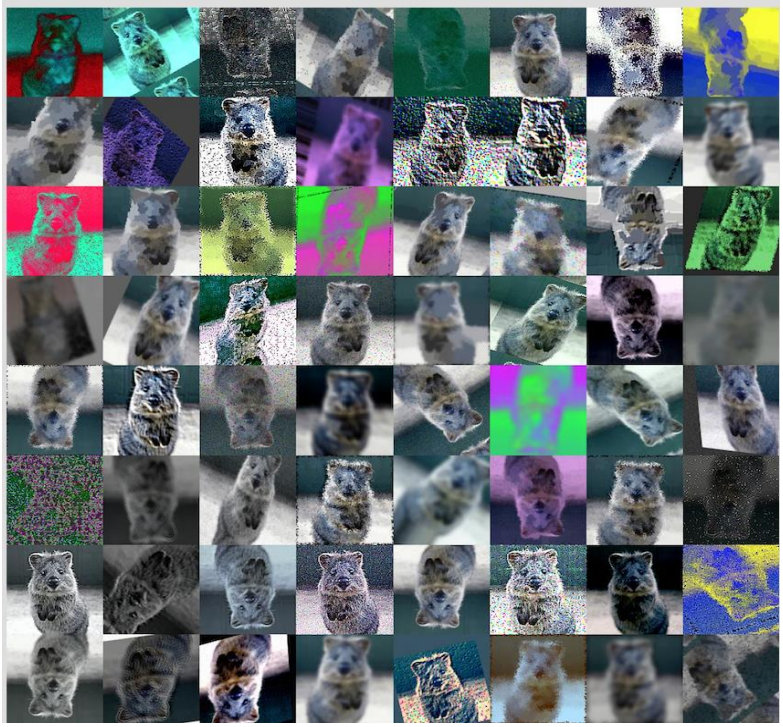
```
entrenamiento_datagen = ImageDataGenerator(
      rescale = 1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest',
      data_format='channels_last',
      brightness_range=[0.5, 2.5])

test_datagen = ImageDataGenerator(rescale=1. / 255)
```

Nota. En el gráfico se presenta la función que realizar el proceso de Data Augmentation, junto con las variaciones a las que es sometida cada imagen.

Con esta práctica el modelo es capaz de ver datos más diversificados, en un conjunto de datos muy pequeño.

Figura 65

Variaciones producidas al aplicar Data Augmentation

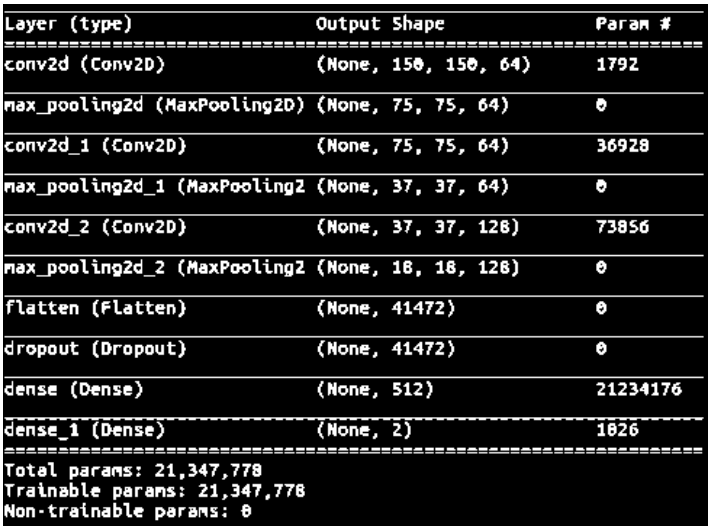


Nota. En el gráfico muestra las variaciones que se pueden generar en una sola imagen. Tomado de *CS231n: Convolutional Neural Networks for Visual Recognition*, (Stanford, 2020).

3. Creación de la arquitectura de la *red neuronal convolucional* de la Figura 66.

Figura 66

Arquitectura de la red neuronal convolucional implementada



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 75, 75, 64)	0
conv2d_1 (Conv2D)	(None, 75, 75, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 37, 37, 64)	0
conv2d_2 (Conv2D)	(None, 37, 37, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 18, 18, 128)	0
flatten (Flatten)	(None, 41472)	0
dropout (Dropout)	(None, 41472)	0
dense (Dense)	(None, 512)	21234176
dense_1 (Dense)	(None, 2)	1026

Total params: 21,347,778
Trainable params: 21,347,778
Non-trainable params: 0

Nota. En el gráfico muestra cómo va ir estructurada la red neuronal convolucional.

Procedimiento:

- Delimitación del tamaño que tomará la imagen de entrada de 150x150.
- Creación de 3 *capas convolucionales* con 64 y 128 filtros de 3x3, enfocados en extraer características de la imagen.
- Funciones de activación *Rectified Linear Unit (ReLU)* para cada capa convolucional.
- Una capa *MaxPooling* de 2x2 encargada de reescalar la imagen y resaltar las características más importantes.
- Una capa *Flatten*, cuya función es generar un vector de características.
- Una capa *Dropout* para analizar cada neurona y eliminar aquellas que se están sobreajustando.

- Por último, dos capas *Dense*, que representa capas de neuronas totalmente conectadas con una función *ReLu* al final de la red, responsable de generación de las ponderaciones para cada clase.

Figura 67

Declaración de la red neuronal convolucional implementada

```
cnn = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150x150 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3), padding='same'),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', padding='same'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu', padding='same'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.4),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(clases)
])
```

Nota. En el gráfico se presenta la función con las declaraciones y configuraciones para cada capa convolucional.

4. Funciones para compilación y entrenamiento, Figura 68: Se establece la función de costo *croos entropy*, seguido de un optimizador con un método más avanzado que el *descenso del gradiente* llamado *Adam*, que ajustará los pesos de la red durante el entrenamiento con el fin de llegar a un mino local que representa la menor de pérdida. Por último, se añade la métrica *accuracy* para el monitoreo de los pasos de entrenamiento y pruebas.

Figura 68

Parámetros de optimización para el entrenamiento

```
cnn.compile(loss='categorical_crossentropy',
            optimizer=optimizers.Adam(lr=lr),
            metrics=['accuracy'])"""

cnn.fit_generator(
            entrenamiento_generador,
            steps_per_epoch=pasos,
            epochs=epocas,
            validation_data=validacion_generador,
            validation_steps=validation_steps,
            verbose=1, callbacks=[callbacks])
```

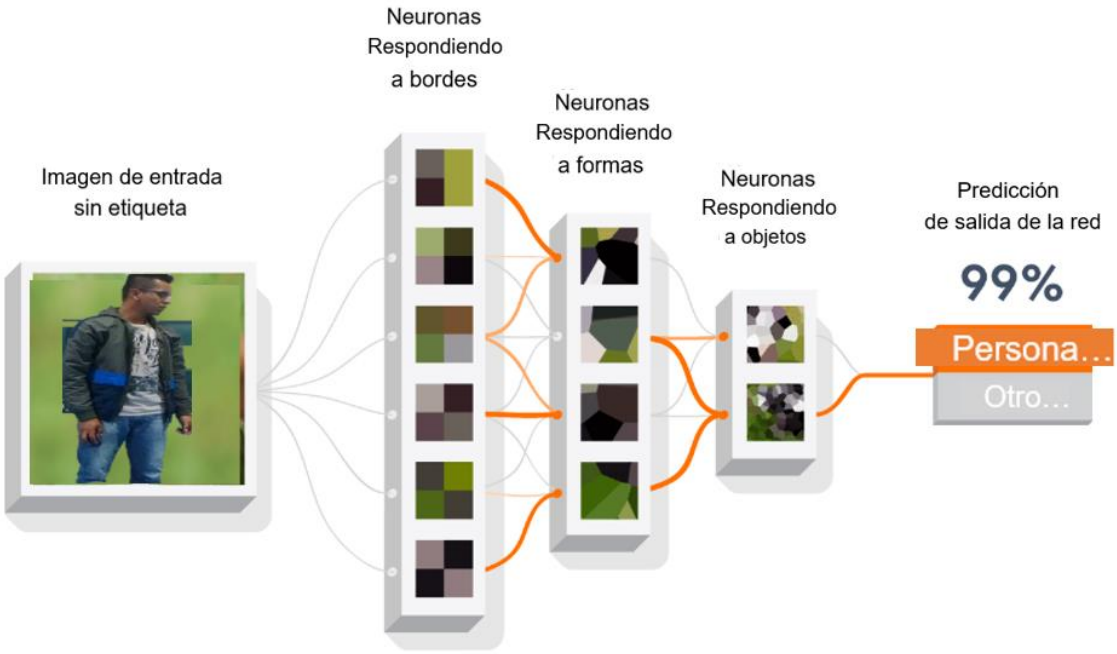
Nota. En el gráfico se presenta las declaraciones de la función de optimización y la función encargada del entrenamiento de la red.

Finalmente, el Dataset de 50 imágenes es dividido según la regla de 80/20, en un conjunto de 40 datos para “train” y 10 para “test o validación”.

Con estas premisas el modelo está listo para el entrenamiento, que inicia con la ejecución de la función “cnn.fit_generator()”. Con ello se genera un modelo entrenado y listo para identificar la persona que es objetivo de la búsqueda.

Figura 69

Anatomía resumida del modelo de clasificación



Nota. En el gráfico muestra como cada capa en una red neuronal está enfocada en analizar distintas características en la imagen.

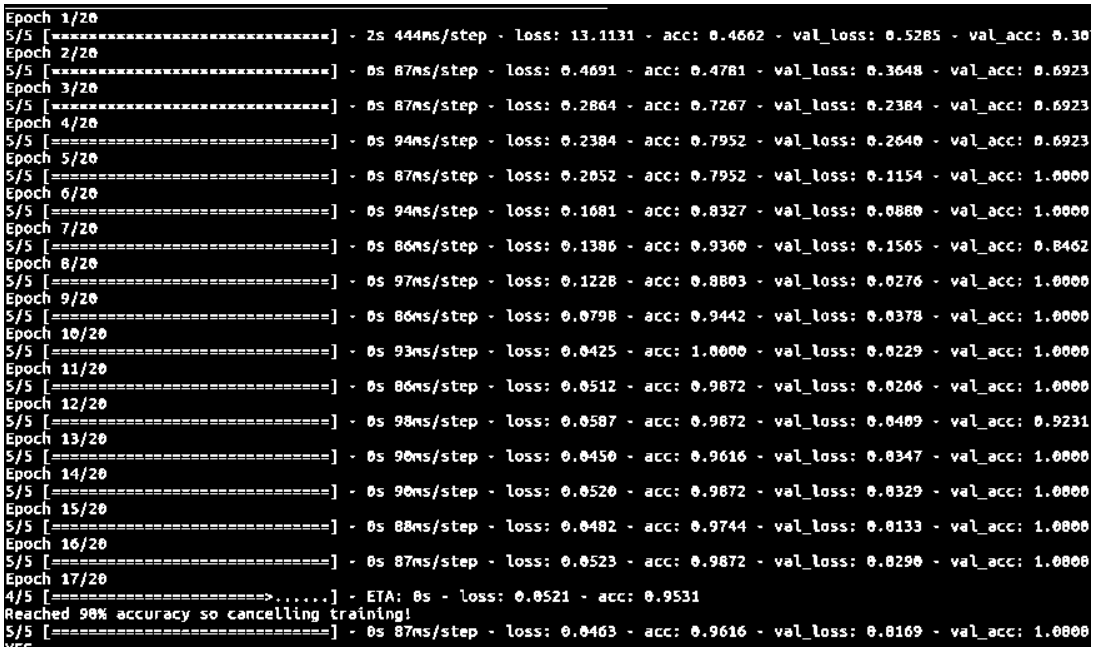
Resultados del entrenamiento

Si bien es cierto para un correcto entrenamiento es necesario disponer de un Dataset de mínimo 500 imágenes que en efecto tomaría más de una hora su procesamiento. Esto significaría capturar toda esa cantidad de imágenes de la persona seleccionada para el seguimiento y dejarla en espera junto con el *micro-UAV* volando todo ese intervalo de tiempo hasta terminar el entrenamiento, lo cual no es una opción adecuada, teniendo también en cuenta que la autonomía de vuelo es una limitante, de manera que mucho antes de acabar el entrenamiento el *micro-UAV* habrá abortado la misión.

Ante ello, la acción a tomar fue sacrificar la calidad del entrenamiento cuyos resultados se muestran en la Figura 70, de modo que el Dataset cuente con tan solo 50 imágenes, que representa el 10% del valor mínimo recomendado, además de una reducción en el número de *épocas* ejecutadas junto con otros parámetros que permitieron reducir el tiempo de entrenamiento del modelo de clasificación creado en el apartado anterior, siendo este tiempo de espera de no más allá de 10 segundos.

Figura 70

Resultados obtenidos en una prueba del entrenamiento del modelo



Nota. En el gráfico muestra a detalle los valores de perdida y precisión en cada época del entrenamiento de la red.

Resultados:

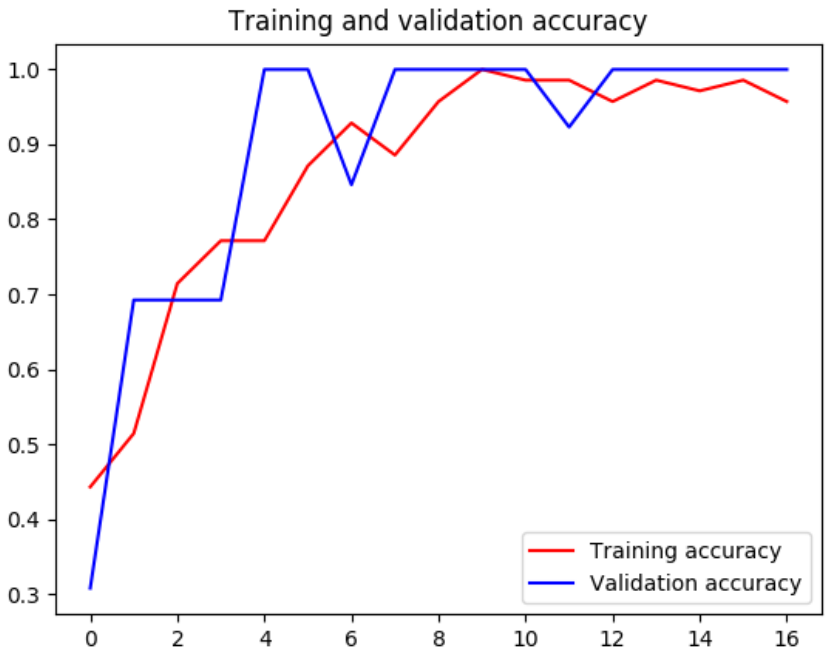
- Perdida en el dataset de entrenamiento: $loss = 0.043$
- Precisión en el dataset de entrenamiento: $acc = 0.9616$

- Perdida en el dataset de validación o test: $val\_loss = 0.8169$
- Precisión en el dataset de validación o test: $val\_acc = 1.0000$

En consecuencia, como se aprecia en la Figura 71, el modelo entrenado se degrada en cierta forma, lo cual conlleva a una convergencia muy rápida que se traduce a un *sobreajuste (overfitting)* en el entrenamiento y a una afectación en la capacidad de generalización de su conocimiento, es decir el modelo reconoce a la persona siempre y cuando esta se encuentre en un ambiente con condiciones similares de luminosidad a las de las fotos capturadas inicialmente.

Figura 71

Precisión de obtenida en una prueba del entrenamiento del modelo

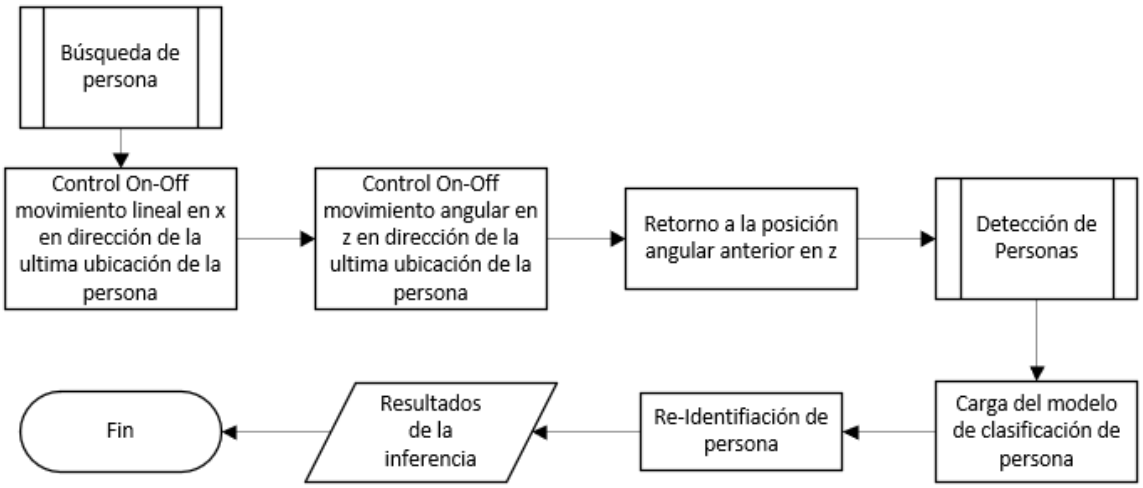


Nota. El gráfico muestra los resultados de la precisión del modelo tanto para los datos de entrenamiento y validación en cada época.

El proceso de ejecución de esta técnica se ilustra en la Figura 72, que representa el diagrama de flujo del algoritmo de Re-Identificación.

Figura 72

Diagrama de flujo: Búsqueda y Re-Identificación de persona



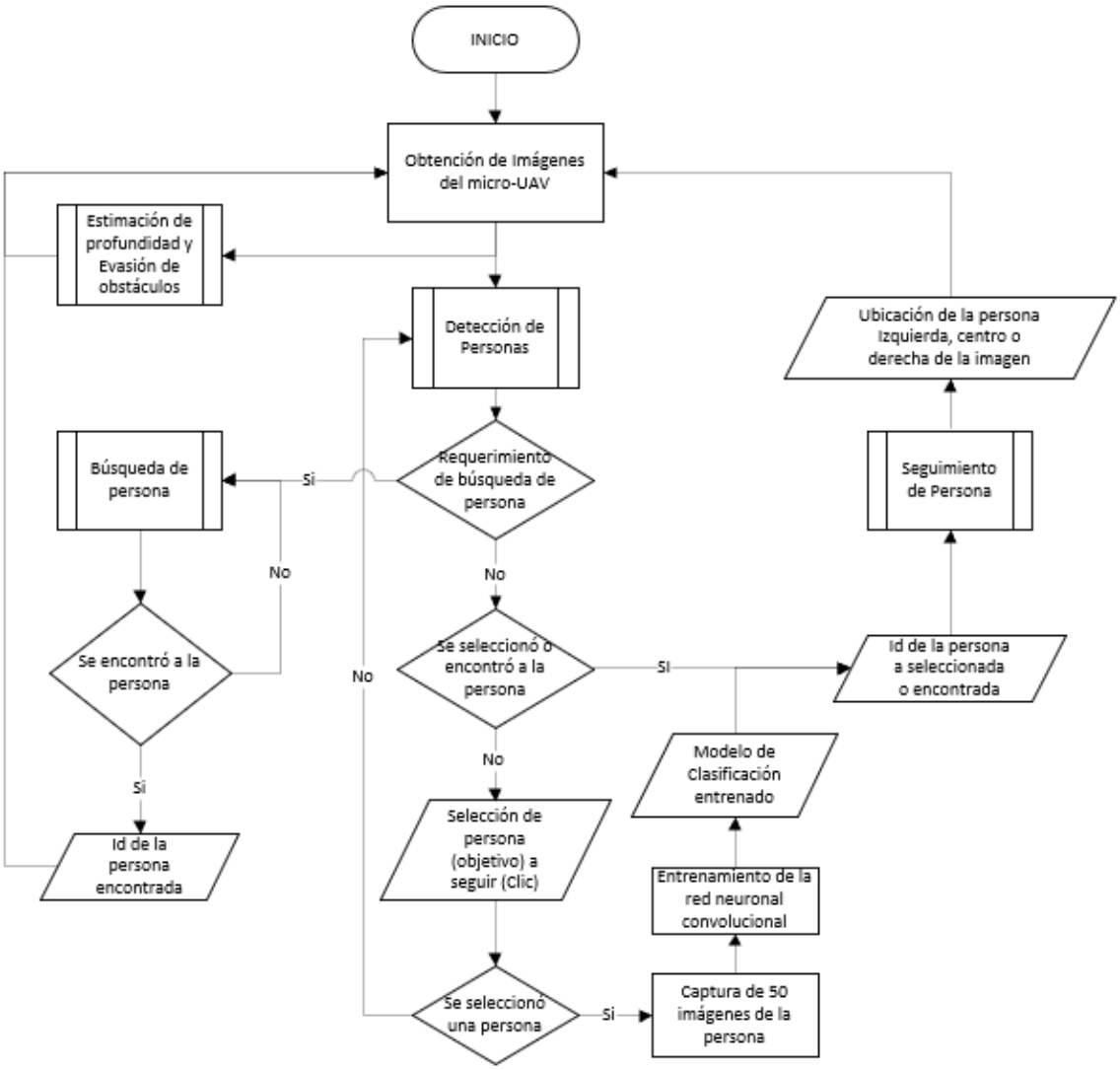
Nota. El gráfico corresponde al diagrama de flujo de la secuencia que debe seguir el proceso de búsqueda y re-identificación de persona.

Diagrama de flujo del sistema completo

Este diagrama integra todas las etapas del sistema implementado en la estación en tierra, mostrando el flujo del proceso en general.

Figura 73

Diagrama de flujo del sistema completo



Nota. El gráfico corresponde al diagrama de flujo del sistema en conjunto: Detección, seguimiento, evasión de obstáculos, búsqueda y Re-Identificación de persona

Capítulo VI

Pruebas experimentales y Resultados

Las evaluaciones del desempeño del sistema de detección, seguimiento, evasión de obstáculos y búsqueda, son realizadas en distintos escenarios, en situaciones con distintas condiciones de viento, luminosidad y cantidad obstáculos. Para ello se ha considerado dos entornos en los que se llevaran a cabo, el primero corresponde al interior de una vivienda con iluminación artificial y el segundo corresponde a un patio exterior al aire libre con iluminación ambiental.

Es importante recalcar que el modelo de detección de objetos YOLOv4 se ha configurado para que detecte personas con porcentajes de certeza superiores al 60% para evitar falsos positivos (reconocer a otros objetos como personas).

Evaluaciones en Entorno 1 - Interior de una vivienda

Estas primeras evaluaciones fueron hechas al interior de una vivienda de 8.5 metros de ancho y 16 metros de largo, siendo este un ambiente controlado, en el que podemos variar las condiciones de luz, ya sea abriendo ventanas, encendiendo o apagando lámparas. Además, hay la posibilidad de agregar obstáculos en distintas posiciones.

Pruebas de detección de personas a diferentes distancias

La primera prueba es analizar la precisión del modelo de detección YOLOv4 sometido a 3 distintas luminosidades y distancias, con ello podemos tener un estimado de si es capaz de detectar personas a distancias lejanas a la que se encuentra el *micro-UAV* en un ambiente interno o cerrado.

Tabla 8

Entorno 1 - Porcentajes de certeza en la detección de personas

Prueba	Iluminación (lux)	Porcentaje de certeza a diferentes distancias			
		2 metros	5 metros	10 metros	15 metros
1	50	99.9 - 99.6%	99.7 - 92.3%	99.3 - 90.1%	97.6 - 89.0%
2	210	99.9 - 99.8%	99.8 - 98.3%	99.4 - 98.1%	98.2 - 89.2%
3	350	100.0 - 99.8%	99.7 – 99.2%	99.7 – 98.5%	98.2 – 91.4%

Nota. Esta tabla muestra los porcentajes de certeza al detectar personas a 4 distancias diferentes.

Figura 74

Interior de la vivienda a distintas condiciones de Luz

Nota. El gráfico muestra las 3 distintas condiciones de luz en las que se realizaron las pruebas.

De los resultados obtenidos de la Tabla 8 se puede notar como en un ambiente controlado en el que se mantienen las mismas condiciones de luz a distintas distancias, el modelo puede ser capaz reconocer personas hasta los 15 metros y con muy buenos valores de certeza, es decir los valores de que tan seguro está el modelo sobre si el objeto encontrado es una persona. Estos resultados que se muestran en cada columna presentan dos valores de certeza, los valores más altos de la izquierda corresponden a la persona con camiseta blanca, mientras que los resultados más bajos de la derecha son

de la persona que esta con vestimenta más oscura. Con ello podemos notar también como influyen vestimentas claras y oscuras en la precisión del modelo.

Figura 75

Entorno 1 - Detección de personas a distintas distancias a 50 lux

Nota. El gráfico muestra las capturas de las inferencias hechas mientras el *micro-UAV* se mantenía en vuelo, estas imágenes corresponden a una luminosidad de 50 lux, que a pesar de ser baja presenta resultados muy buenos.

Un punto importante a tener en cuenta es que, a partir de los 10 metros, el modelo demuestra falsos negativos (no reconoce personas) en 5 o 6 de cada 20 *frames*. Si bien es cierto la iluminación influye bastante, pero en mayor parte es debido a la distancia, ya que las personas se ven bastante pequeñas y esto crea cierta confusión para el modelo al momento de la detección. Importante mencionar que este comportamiento va repercutir mucho en el rastreo.

Pruebas de rastreo de la persona objetivo discriminando otras

El objetivo de esta prueba es verificar hasta que distancia de separación el sistema MOT logra mantener el rastreo de la persona seleccionada en la imagen, mientras el *micro-UAV* se mantiene volando en una posición fija dentro de un ambiente interno o cerrado.

Tabla 9

Entorno 1 - Rastreo de la persona seleccionada a diferentes distancias

Prueba	Iluminación (lux)	Permanece o no el rastreo a diferentes distancias			
		2 metros	5 metros	10 metros	15 metros
1	50	Si	Si	Si	No
2	210	Si	Si	Si	No
3	350	Si	Si	Si	No

Nota. Esta tabla muestra si el rastreo de la persona seleccionada permaneció a 4 diferentes distancias.

En la Tabla 9 se puede notar como afectan los falsos negativos (mencionados en la sección anterior) al momento de mantener el rastreo, manteniéndolo hasta una distancia de separación máxima del *micro-UAV* de 10 metros. Esto es debido a que el sistema *MOT* se alimenta de las detecciones generadas por *YOLOv4* en cada frame, pero en el caso de no haberlas, el sistema *MOT* con el *Filtro de Kalman* predice hasta 4 detecciones posteriores, es decir hasta 4 frames, pero si no hay detecciones en mayor cantidad de frames, el sistema da por perdida a la persona, elimina su id y en consecuencia deja de rastrearla.

Figura 76

Entorno 1 - Rastreo de la persona seleccionada a distintas distancias

Nota. El gráfico muestra las capturas de realizadas durante el rastreo de la persona al interior de una vivienda con una luminosidad de a 350 lux.

Ventajosamente existe una solución ante el inconveniente de la pérdida del rastreo a más de 10 metros. Esta se basa en modificar ciertos parámetros en el sistema *MOT* para que intente predecir las detecciones por mayor cantidad de *frames*, no es tan precisa, pero funciona en cierta manera. Para este trabajo de investigación se mantendrá las predicciones hasta 4 *frames* debido a que el seguimiento y rastreo no se harán a distancias mayores a 5 metros de separación entre la persona y el *micro-UAV.*

Pruebas de seguimiento de la persona en trayectoria sin obstáculos

Estas pruebas son realizadas en una trayectoria libre de obstáculos lo que posibilita aumentar la velocidad de caminata de la persona y comprobar el funcionamiento del sistema de seguimiento sin comprometer la integridad del *micro-UAV* ante posibles colisiones con obstáculos. Es importante considerar que partir de los 1.5 m/s ocurre la transición en que la persona pasa de caminar a correr, para lo que se necesitaría de un lugar externo, abierto y mucho más grande para realizar pruebas a velocidades mayores.

Tabla 10

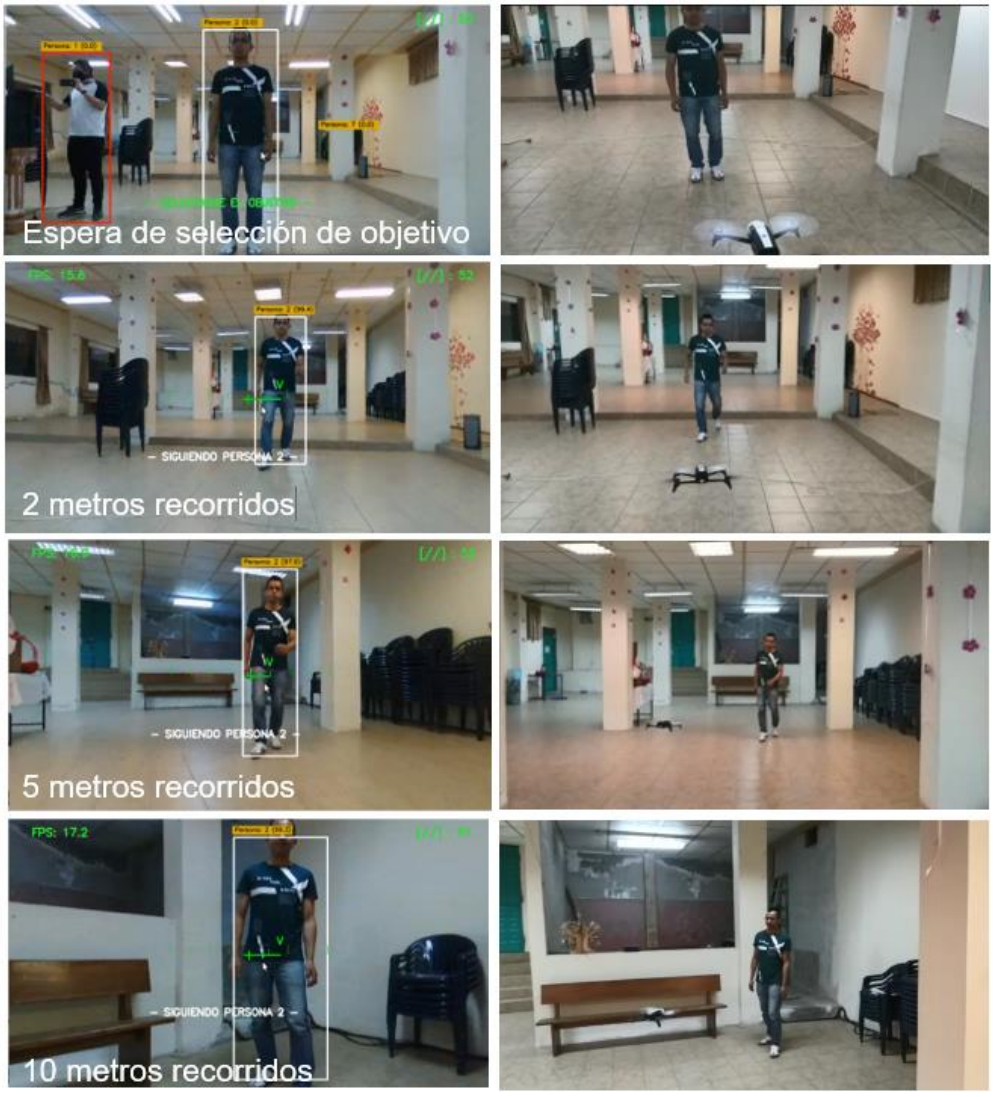
Entorno 1 - Seguimiento en trayectoria aleatoria sin obstáculos

Prueba	**Velocidad de caminata de la persona**	**Estado a diferentes distancias**	
		10 metros	**15 metros**
1	0.5 m/s (lento)	Completado	Completado
2	1 m/s (moderado)	Completado	Completado
3	1.5 m/s (apresurado)	Completada	Completado

Nota. Esta tabla muestra si el seguimiento a la persona seleccionada fue completado en un tramo de 10 y 15 metros a 3 distintas velocidades en el interior de una vivienda.

Figura 77

Entorno 1 - Seguimiento de persona en trayectoria sin obstáculos



Nota. El gráfico muestra las capturas hechas durante el seguimiento de la persona seleccionada, en el interior de una vivienda, a una velocidad de 1.5 m/s, a 350 lux, durante 10 metros de recorrido.

Pruebas de seguimiento de la persona con evasión de obstáculos

Para no comprometer la integridad del *micro-UAV* en estas pruebas con obstáculos se ha decidido omitir la velocidad de caminata de 1.5 m/s puesto que esta genera velocidades altas de desplazamiento en el *micro-UAV* quedando susceptible ante colisiones. Si bien es cierto se cuenta con un modelo de estimación de profundidad para evadir obstáculos, pero cabe recalcar que no es muy robusto para trabajar a altas velocidades.

Tabla 11

Entorno 1 - Seguimiento y evasión de obstáculos en trayectoria aleatoria

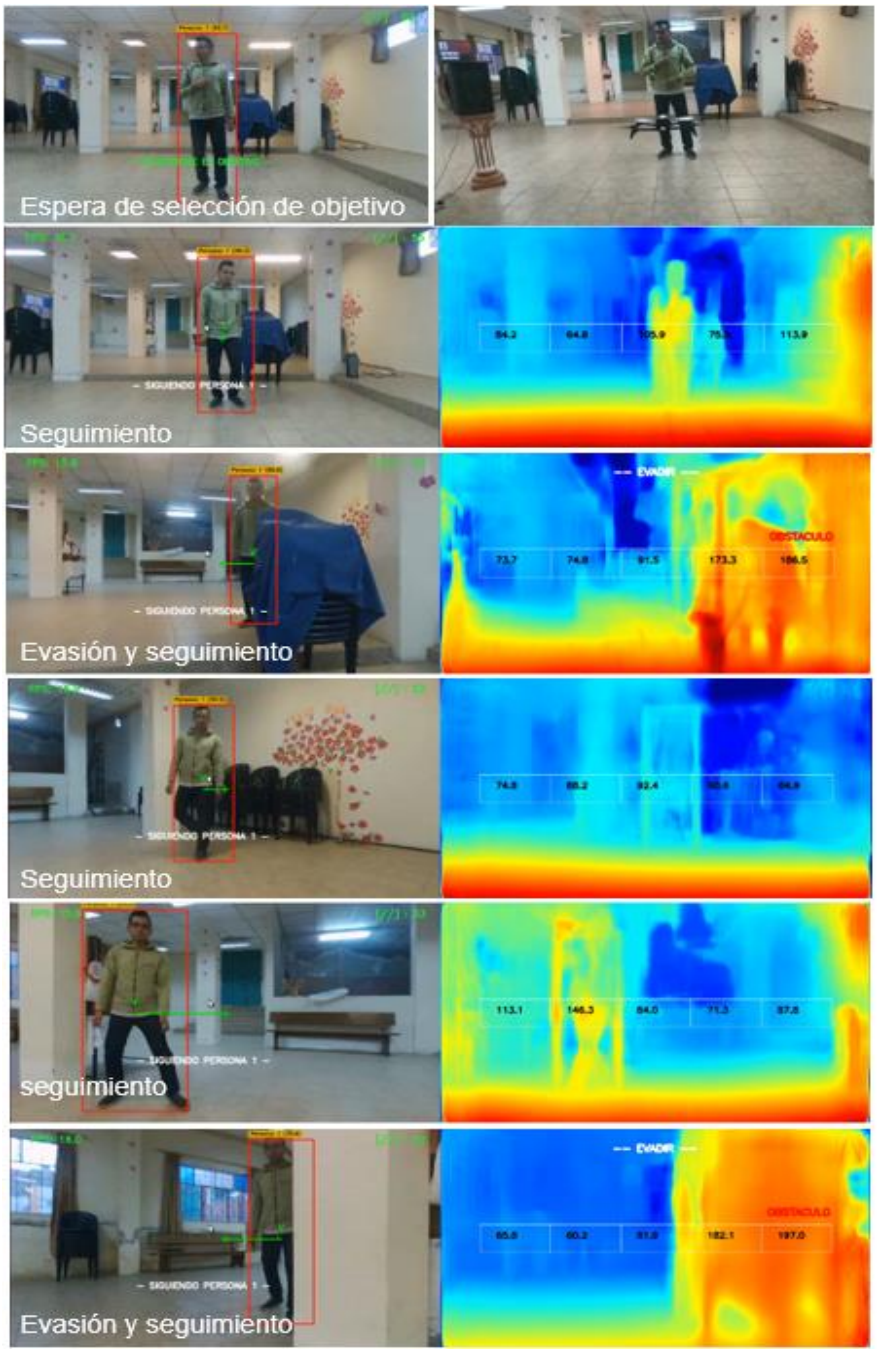
Prueba	Iluminación (lux)	Estado a diferentes velocidades de caminata de la persona	
		0.5 m/s	1 m/s
1	50	Completado	Completado
2	210	Completado	Completado
3	350	Completado	Completado

Nota. Esta tabla muestra si el seguimiento a la persona seleccionada fue completado en un tramo de 10 metros con obstáculos, a dos velocidades distintas y con cambios de iluminación al interior de una vivienda.

Las pruebas evasión de obstáculos se realizaron a la más alta condición de luminosidad en el interior de la vivienda y con obstáculos separados a una distancia mínima de 2 metros. Estas fueron culminadas con éxito, 3 pruebas a una velocidad de caminata lenta (0.5 m/s) y 3 pruebas a una velocidad moderada (1 m/s), a 350 lux, recorriendo una trayectoria de 10 metros como se observa a continuación.

Figura 78

Entorno 1 - Seguimiento de persona en trayectoria con obstáculos



Nota. El gráfico muestra las capturas hechas durante el seguimiento de la persona en una trayectoria con obstáculos al interior de una vivienda y a una velocidad de 1 m/s.

Pruebas de búsqueda y re-identificación de persona

Las siguientes pruebas corresponden a la implementación del sistema completo, en el cual el algoritmo sigue a la persona seleccionada evadiendo varios obstáculos. En el transcurso de la trayectoria, la persona va ser obstruida por cualquier objeto en el entorno, de modo que el *micro-UAV* perderá su rastro y esto dará inicio a la etapa de búsqueda y re-identificación. En esta etapa el objetivo es que mientras el *micro-UAV* se desplaza en busca de su objetivo, el sistema debe ser capaz de discriminar cualquier otra persona que encuentre en el camino y únicamente reconocer a la persona anteriormente seleccionada. Los resultados de las pruebas se muestran en la siguiente tabla.

Tabla 12

Entorno 1 - Búsqueda y Re-Identificación en trayectorias aleatorias

Prueba	Estado de Re-Identificación de persona	Porcentaje de certeza de la persona Re-Identificada con el modelo entrenado	Falsos	
			Positivos	**Negativos**
1	Encontrada	75.90 %	No	No
2	Encontrada	80.04%	Si	No
3	Encontrada	83.35%	No	No
4	Encontrada	60.31%	No	Si
5	Encontrada	68.00%	Si	No
6	Encontrada	77.01%	No	Si
7	Encontrada	83.27%	Si	No

Nota. Esta tabla muestra los resultados de certeza obtenidos al re-identificar a la persona y a la vez muestra si se produjeron falsos positivos y negativos durante este proceso al interior de una vivienda.

En la Tabla 12 es notorio que el sistema en todas las pruebas logra encontrar a una persona, pero no en todos los casos es la persona correcta. Solamente en el 57% de las pruebas realizadas logró el objetivo y estas corresponden a las filas sombreadas.

Figura 79

Entorno 1 - Pruebas de todo el sistema en conjunto.

Nota. El gráfico muestra las capturas hechas durante proceso de Detección, seguimiento, evasión de obstáculos, búsqueda y re-identificación de persona en una trayectoria con obstáculos al interior de una vivienda.

Algunas de estas pruebas presentaron falsos negativos, lo que significa que una vez que la persona objetivo apareció a la vista del *micro-UAV*, el algoritmo logró reconocer a la persona, pero después de varios *frames*. Las pruebas que no presentan falsos positivos quieren decir que la re-identificación fue inmediata.

Las pruebas en las que ocurrieron falsos positivos corresponden una mala re-identificación de persona, a causa de cambios de luz o ciertas vestimentas similares, pero principalmente gran parte de los problemas es debido al modelo entrenado, que como se explicó en el capítulo V, no es tan eficiente, a raíz del poco tiempo y de las pocas imágenes usadas para su entrenamiento.

Evaluaciones en Entorno 2 - Patio exterior al aire libre

Estas evaluaciones tienen lugar en un patio exterior al aire libre de 9.0 metros de ancho y 19 metros de largo, siendo este un ambiente parcialmente controlado, en el que podemos variar la disposición de los obstáculos. Pero el entorno es afectado por cambios repentinos de luz, sombras, ráfagas de viento, entre otros.

Pruebas de detección de personas a diferentes distancias

De igual manera que en el entorno anterior se realizaron pruebas para evaluar la precisión del modelo de detección YOLOv4 sometido a 3 distintas luminosidades y distancias de separación del *micro-UAV* en un ambiente externo.

Tabla 13

Entorno 2 - Porcentajes de certeza en la detección de personas

Prueba	Iluminación (lux)	Porcentaje de certeza a diferentes distancias			
		2 metros	5 metros	10 metros	15 metros
1	690	99.8 - 99.0%	99.4 - 95.8%	98.9 - 87.1%	97.4 - 80.1%
2	1060	99.9 - 99.6%	99.6 - 98.9%	99.3 - 98.5%	99.1 - 92.4%
3	4661	99.8 - 99.5%	99.1 – 98.4%	99.2 – 95.6%	98.3 – 82.0%

Nota. Esta tabla muestra los porcentajes de certeza al detectar personas a 4 distancias diferentes.

Figura 80

Patio exterior a distintas condiciones de luz

Nota. El gráfico muestra capturas del Entorno 2 a distintas condiciones de luz. La iluminación de 690 lux corresponde a un ambiente al aire libre, con sombra. A 1060 lux se trata de un ambiente parcialmente con sobra y luz solar. A 4661 un ambiente al medio día completamente iluminado con luz solar.

De los resultados obtenidos de la Tabla 13 se puede notar como en un ambiente en el que las condiciones de luz no pueden ser controladas, el modelo es también capaz de reconocer personas hasta los 15 metros, con buenos valores de certeza, pero en comparación a los de un entorno interior controlado son un poco más bajos debido los factores externos y a la claridad de la vestimenta como se había mencionado. Los valores

más altos de la izquierda corresponden a la persona con camiseta blanca, mientras que los resultados más bajos de la derecha son de la persona que esta con vestimenta más oscura.

Figura 81

Entorno 2 - Detección de personas a distintas distancias a 4661 lux

Nota. El gráfico muestra las capturas de las inferencias hechas mientras el *micro-UAV* se mantenía en vuelo, estas imágenes corresponden a una luminosidad de 4661 lux.

Al igual que en las pruebas en el interior de la vivienda, a partir de los 10 metros, el modelo demuestra falsos negativos (no reconoce personas) en 5 o 6 de cada 20 frames y de igual forma va influir este comportamiento en el proceso de rastreo.

Pruebas de rastreo de la persona objetivo discriminando otras

El objetivo de esta prueba es verificar hasta que distancia de separación del *micro-UAV* el sistema MOT logra mantener el rastreo de la persona seleccionada en la imagen, mientras el *micro-UAV* se mantiene volando en una posición fija y en un ambiente externo.

Tabla 14

Entorno 2 - Rastreo de la persona seleccionada a diferentes distancias

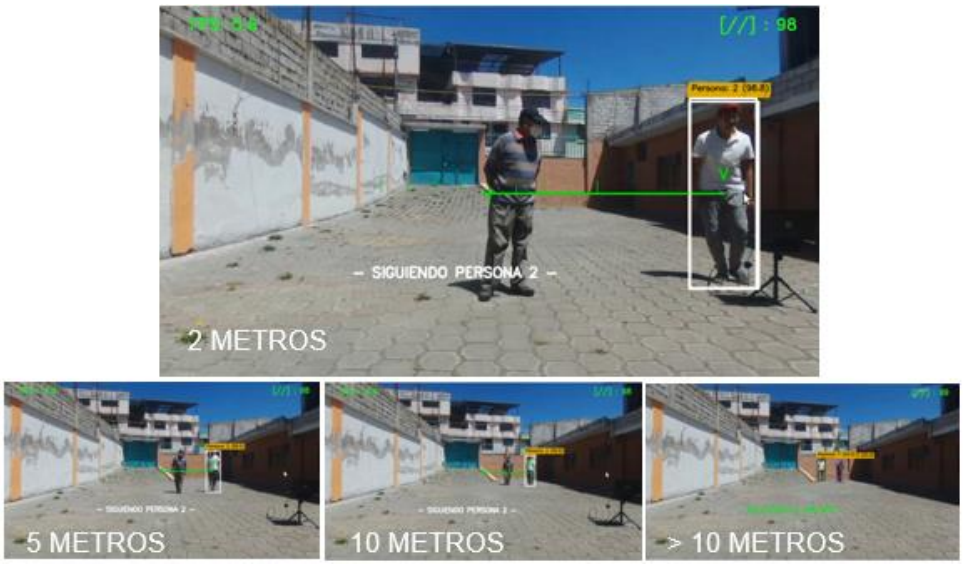
Prueba	Iluminación (lux)	Permanece o no el rastreo a diferentes distancias			
		2 metros	5 metros	10 metros	15 metros
1	690	Si	Si	No	No
2	1060	Si	Si	Si	No
3	4661	Si	Si	Si	No

Nota. Esta tabla muestra si el rastreo de la persona seleccionada permaneció a 4 diferentes distancias.

En la Tabla 14 se puede notar como ambientes exteriores también afectan los falsos negativos al momento de mantener el rastreo, manteniéndolo hasta una distancia máxima de 10 metros, pero menores a 10 metros en el caso de un ambiente con sombra de 690 lux de luminosidad.

Figura 82

Entorno 2 - Rastreo de la persona seleccionada a distintas distancias



Nota. El gráfico muestra las capturas de realizadas durante el rastreo de la persona en un patio exterior con una luminosidad de a 4661 lux.

En vista de que el terreno no es muy grande, tampoco se ejecutará ninguna modificación en el algoritmo para ambientes externos y se mantendrá las predicciones hasta 4 *frames* para realizar un seguimiento y rastreo a distancias menores o iguales a 5 metros de separación entre la persona y el *micro-UAV.*

Pruebas de seguimiento de la persona en trayectoria sin obstáculos

Estas pruebas son realizadas en una trayectoria libre de obstáculos y en un ambiente exterior, lo que posibilita aumentar la velocidad de caminata de la persona y comprobar el funcionamiento del sistema de seguimiento sin comprometer la integridad del *micro-UAV* ante posibles colisiones con obstáculos. Para el seguimiento de personas mientras esta trota o corre, es recomendable un ambiente externo, pero mucho más grande.

Tabla 15

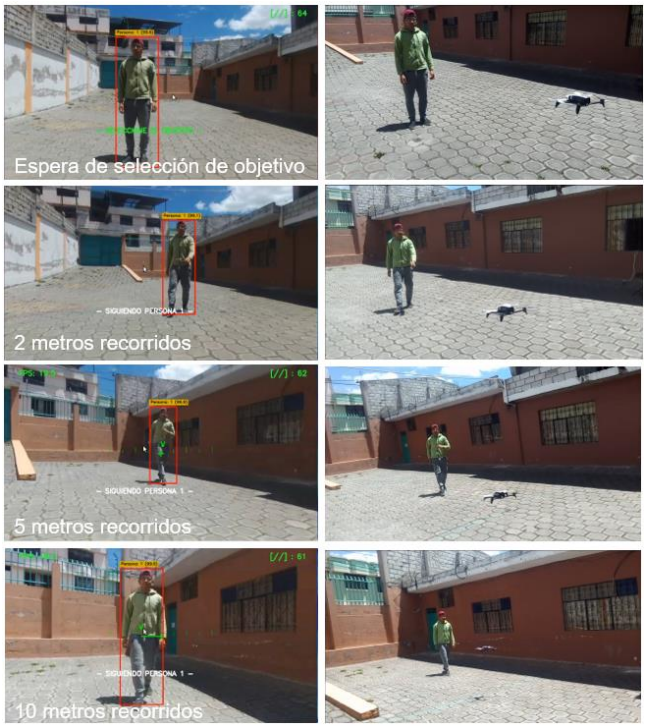
Entorno 2 - Seguimiento en trayectoria aleatoria sin obstáculos

Prueba	Velocidad de caminata de la persona	Estado a diferentes distancias	
		10 metros	15 metros
1	0.5 m/s (normal)	Completado	Completado
2	1 m/s (moderado)	Completado	Completado
3	1.5 m/s (apresurado)	Completada	Completado

Nota. Esta tabla muestra si el seguimiento a la persona seleccionada fue completado en un tramo de 10 y 15 metros a 3 distintas velocidades en un patio exterior.

Figura 83

Entorno 2 - Seguimiento de persona en trayectoria sin obstáculos



Nota. El gráfico muestra las capturas hechas durante el seguimiento de la persona seleccionada, en un patio exterior, a una velocidad de 1.5 m/s, a 4661 lux, durante 10 metros recorridos.

Pruebas de seguimiento de la persona con evasión de obstáculos

Para no comprometer la integridad del *micro-UAV* en estas pruebas con obstáculos se ha decidido omitir la velocidad de caminata de 1.5 m/s puesto que esta genera velocidades altas de desplazamiento en el *micro-UAV* quedando susceptible ante colisiones.

Tabla 16

Entorno 2 - Seguimiento y evasión de obstáculos en trayectoria aleatoria

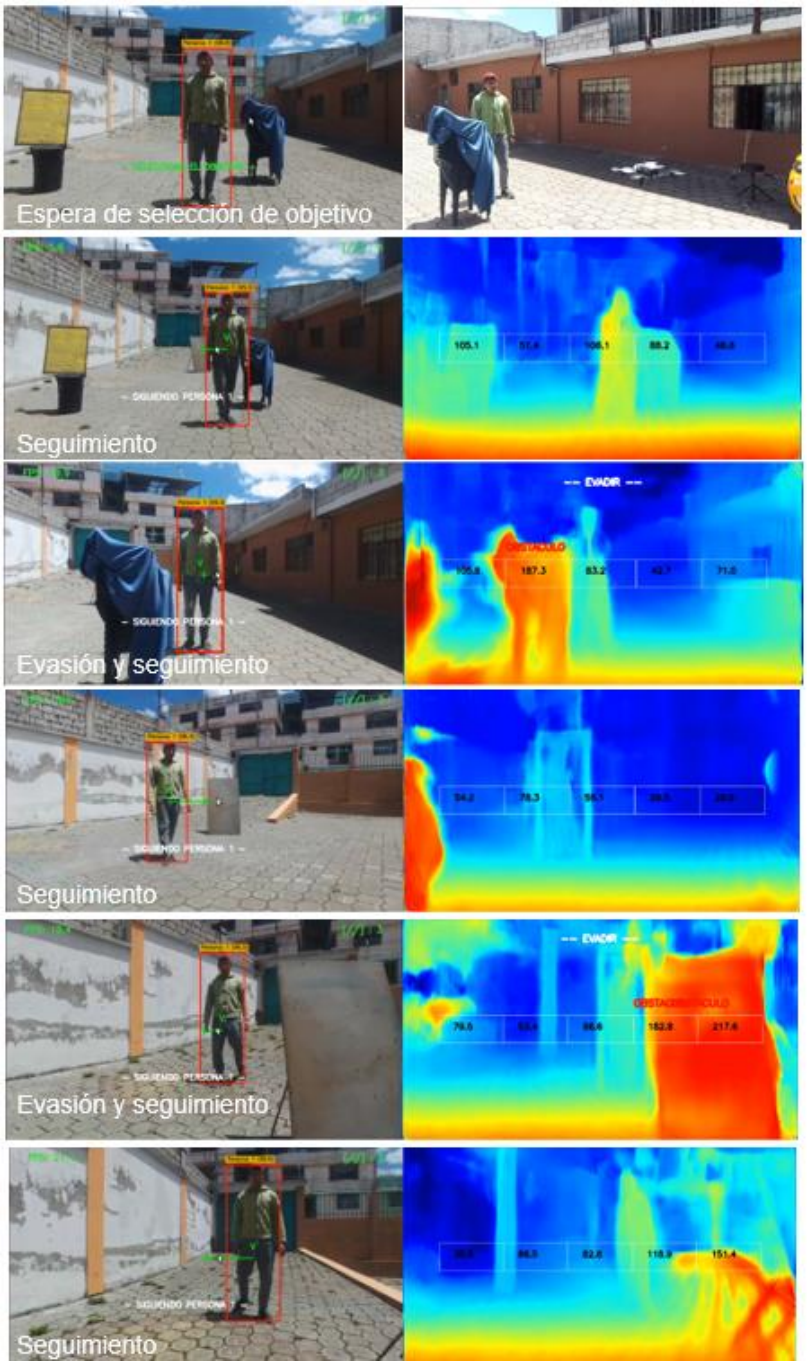
Prueba	Iluminación (lux)	Estado a diferentes velocidades de caminata de la persona	
		0.5 m/s	1 m/s
1	690	Completado	Completado
2	1060	Completado	Completado
3	4661	Completado	Completado

Nota. Esta tabla muestra si el seguimiento a la persona seleccionada fue completado en un tramo de 10 metros con obstáculos, a dos velocidades distintas y con cambios de iluminación en un patio exterior.

Las pruebas evasión de obstáculos se realizaron a la más alta condición de luminosidad en un patio exterior y con obstáculos separados a distancias mayores a 2 metros. Estas fueron culminadas con éxito, 3 pruebas a una velocidad de caminata lenta (0.5 m/s) y 3 pruebas a una velocidad moderada (1 m/s), a 4661 lux, recorriendo una trayectoria de 10 metros como se observa a continuación.

Figura 84

Entorno 2 - Seguimiento de persona en trayectoria con obstáculos



Nota. El gráfico muestra las capturas hechas durante el seguimiento de la persona en una trayectoria con obstáculos en un patio exterior y a una velocidad de 1 m/s.

Pruebas de búsqueda y re-identificación de persona

En esta etapa el objetivo es que mientras el *micro-UAV* se desplaza en el patio exterior en busca de su objetivo una vez que lo perdió de vista, el sistema debe ser capaz de discriminar cualquier otra persona que encuentre en el camino y únicamente reconocer a la persona anteriormente seleccionada. Los resultados de las pruebas se muestran en la siguiente tabla.

Tabla 17

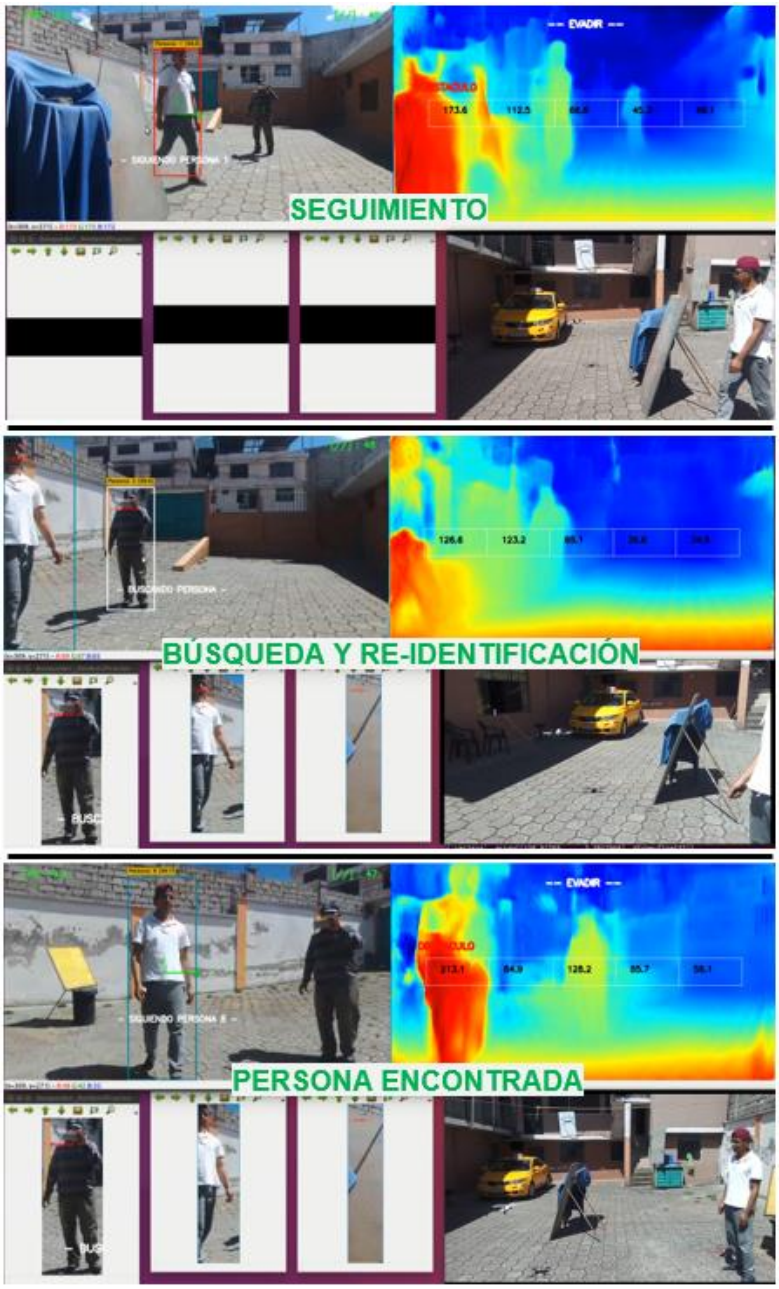
Entorno 2 - Búsqueda y Re-Identificación en trayectorias aleatorias

Prueba	Estado de Re-Identificación de persona	Porcentaje de certeza de la persona Re-Identificada con el modelo entrenado	Falsos	
			Positivos	**Negativos**
1	Encontrada	82.00 %	Si	No
2	Encontrada	94.10%	No	No
3	Encontrada	86.09%	No	Si
4	Encontrada	90.77%	No	Si
5	Encontrada	75.50%	Si	No
6	Encontrada	79.02%	No	Si
7	Encontrada	69.01%	Si	No

Nota. Esta tabla muestra los resultados de certeza obtenidos al re-identificar a la persona y a la vez muestra si se produjeron falsos positivos y negativos durante este proceso en un patio exterior.

Figura 85

Entorno 2 - Pruebas de todo el sistema en conjunto



Nota. El gráfico muestra las capturas hechas durante proceso de detección, seguimiento, evasión de obstáculos, búsqueda y re-identificación de persona.en una trayectoria con obstáculos en un patio exterior.

Al igual que en el interior de una vivienda, el sistema siempre llega a encontrar a una persona, como se muestra en la Tabla 17, pero no en todos los casos es la persona correcta. De igual manera solo el 57% de las pruebas realizadas logró el objetivo y estas corresponden a las filas sombreadas. Se puede notar como en estas pruebas en exteriores se presentaron más falsos negativos como se observa en la Figura 85. No obstante, solo una de las pruebas no tuvo falsos positivos y fue una re-identificación inmediata de la persona buscada.

Las pruebas en las que ocurrieron falsos positivos igualmente corresponden a una mala re-identificación de persona, es decir el *micro-UAV* empieza a seguir a una persona totalmente distinta. La causa de estos falsos negativos y positivos son generalmente por cambios de luz o ciertas vestimentas similares, pero en mayor parte es debido al modelo entrenado.

Capítulo VII

Conclusiones y Recomendaciones

Conclusiones

Tras llevar a cabo una serie de investigaciones sobre técnicas y estrategias de inteligencia artificial enfocadas al área de computer visión, junto con el uso de un micro-UAV y su cámara incorporada, se desarrolló e implementó un sistema autónomo para la detección, el seguimiento y búsqueda de una persona en dos entornos distintos y complejos, correspondientes al interior de una vivienda y a un patio exterior al aire libre. De manera que se emplearon técnicas de estimación de profundidad, rastreo, búsqueda y re-identificación personas, siendo esta ultima un desafió en la que muchos investigadores y empresas apuestan cada vez más a su desarrollo para ser puestas en práctica sobre circuitos cerrados de cámaras de seguridad, abriendo paso a una serie de aplicativos tanto civiles como militares.

El sistema de detección y reconocimiento de objetivos YOLOv4 implementado, ofrece mejores resultados en cuanto a presión y velocidad a diferencia de su antecesor YOLOv3. El sistema presentó resultados en cuanto a porcentajes de certeza entre 100 y 90% en interiores, y el 99 y 80% en exteriores con distintas condiciones de luz, para evaluaciones hasta una distancia de detección de personas de 15 metros y con velocidades de procesamiento de aproximadamente 20FPS en un ordenador con una tarjeta Nvidia GTX-1050Ti. Resultados bastante buenos en comparación a modelos de detección como ssd_mobilenet, ssd_inception, faster_rcnn, entre otros modelos de TensorFlow, cuyas precisiones y velocidades son inferiores.

El sistema de evasión de obstáculos fue implementado gracias a algoritmos de estimación de profundidad en imágenes monoculares, que, a través de un mapa de profundidad, se delimitó las secciones en la que se analizaría si hay obstáculos a 2 metros, distancia adecuada para responder a ellos con anticipación y evitar colisiones en los extremos. Este análisis fue hecho a través de ponderaciones en las intensidades de pixeles, llegando a definirse que. para valores mayores a 160 son considerados obstáculos y cuya acción de evasión es realizada simplemente por un controlador ON-OFF.

Para la etapa de seguimiento, previamente fue realizado el modelamiento de la planta a través de la estimación del movimiento del micro-UAV captado entre cada frame, tanto para el movimiento en pitch (hacia delante y atrás) y en yaw (giro angular sobre el propio eje). A partir de ello tan solo fue necesario el diseño de dos controladores PI. El controlador en pitch es el encargado de mantener el seguimiento de la persona desde una distancia de separación adecuada, y el controlador en yaw se enfoca en mantener a la persona centrada en la imagen.

La implementación de la etapa de seguimiento también estuvo acompañada de un sistema MOT, que, por medio de un Filtro de Kalman, identifica a cada persona con un id y ejecuta un rastreo digital de ella mientras se mantiene dentro de la imagen captada por el micro-UAV. Además, el sistema es capaz de predecir hasta 4 frames posteriores en caso de que YOLOv4 no genere detecciones de la persona.

Las pruebas de seguimiento recorriendo tramos de 10 y 15 metros, a velocidades de caminata de 0.5, 1 y 1.5 m/s en un entorno con y sin obstáculos, tanto en interiores como en exteriores, fueron completadas con éxito en distintas condiciones de luminosidad

y manteniendo una distancia de separación menor a 5 metros entre el micro-UAV y la persona. Esto que indica que el sistema es capaz de realizar un buen seguimiento a distintas velocidades incluso a distancias mayores y con obstáculos de por medio.

El rastreo de la persona en la imagen fue posible hasta una distancia de separación máxima de 10 metros entre ella y el micro-UAV, debido a la perdida de la detección por a falsos negativos en 5 o 6 de cada 20 frames producidos en la detección a distancias mayores. Este comportamiento sucedió en pruebas realizadas en ambos entornos, exterior e interior. La solución a ello es simplemente reconfigurar los parámetros en el filtro de Kalman en el sistema MOT, para que efectué predicciones por mayor cantidad de frames e intente no perder el rastreo de la persona objetivo en el caso que la detección falle.

En la etapa de búsqueda se diseñó un modelo de clasificación de imágenes con el uso del framework Tensorflow y la biblioteca Keras para la re-identificación de la persona. El modelo es entrenado con las imágenes de la persona a seguir en cada ejecución del sistema y su rendimiento fue adecuado en tan solo el 57% de las pruebas realizadas en un entorno interior y exterior, presentando en la mayor parte de ellas falsos positivos y negativos. En conclusión, el bajo rendimiento del modelo es debido al tiempo limitado y a la poca cantidad de datos disponibles para su entrenamiento. Lo que provoca que el modelo se sobreajuste y no generalice su conocimiento al momento de hacer las inferencias.

El sistema es susceptible en un 33% a confundirse en la re-identificación, a causa de cambios de luz en el ambiente, personas con vestimentas similares entre otros. Pero generalmente es en vista de la baja robustez del modelo de entrenamiento.

Recomendaciones

Se recomienda mantener una constante revisión de los componentes del micro-UAV, tales como motores, batería, hélices, sistema de amortiguación en la caída, entre otros, a fin de no tener inconvenientes durante el vuelo y no comprometer la integridad de este.

En el caso de contar con ordenadores con mayores capacidades computacionales a las utilizadas en este proyecto de investigación, se recomienda usar detector YOLOv4 configurado para profesar imágenes de 608 x 608, para obtener una mayor precisión en las inferencias a mayores distancias y disminuir la generación de falsos positivos y negativos. Otra opción es el uso de un detector mucho más robusto, desarrollado por los investigadores de Facebook sobre el framework PyTroch, llamado DETR (DEtection TRansformer).

En cuanto al sistema de rastreo, se recomienda usar un sistema de MOT (Multiple Object Tracking) ya estructurado y más robusto, como DeepSort, FairMOT, JDE, Lif_T, entre otros.

Para la etapa de estimación de profundidad monocular, se recomienda el uso de sistemas mucho más precisos y robustos, como lo son SVS, monoResMatch, SemiDepth, modondepth2 entre otros modelos disponibles desarrollados sobre TensorFlow o PyTorch.

En la etapa de búsqueda y re-identificación de persona, se recomienda usar el mismo modelo diseñado en este trabajo de investigación, pero entrenarlo sobre mayor cantidad de épocas, para lo cual se requiere de un computador mucho más potente, que realice el entrenamiento a mayor velocidad y en la misma cantidad de tiempo. La otra

opción es realizar e ir actualizando el entrenamiento con nuevas capturas mientras el micro-UAV ejecuta el seguimiento de la persona.

Como ultima recomendación en la etapa de búsqueda es el entrenamiento de un modelo de clasificación a través de Transfer Learning. De este modo se puede usar modelos robustos, con capas convolucionales, funciones de activación y demás parámetros diseñados por investigadores y puestos a disposición para su libre uso. Un ejemplo es el modelo VGG16.

Es necesario mencionar que la mayor parte de las recomendaciones sobre técnicas y estrategias propuestas para mejorar el rendimiento del sistema, requieren de un ordenador con altos recursos computacionales.

Referencias

Abdor , J., & Delgado, D. (2017). *Seguimiento de un objetivo móvil aplicando monovisión en el sistema de navegación aérea autónoma utilizando GPU.* Tesis Pregrado, Universidad Piloto de Colombia.

Aguilar, W. G., & Angulo, C. (2012). Compensación de los efectos generados en la imagen por el control de navegación del robot Aibo ERS 7. *VII Congreso de Ciencia y Tecnología ESPE.* Sangolquí, Ecuador.

Aguilar, W. G., & Angulo, C. (2012). Compensación y aprendizaje de efectos generados en la imagen durante el desplazamiento de un robot. *X Simposio CEA de Ingeniería de Control.* Barcelona, Spain.

Aguilar, W. G., & Angulo, C. (2013). Estabilización robusta de vídeo basada en diferencia de nivel de gris. *VIII Congreso de Ciencia y Tecnología ESPE.* Sangolquí, Ecuador.

Aguilar, W. G., & Angulo, C. (2014). Estabilización de vídeo en micro vehículos aéreos y su aplicación en la detección de caras. *IX Congreso de Ciencia y Tecnología ESPE.* Sangolquí, Ecuador.

Aguilar, W. G., & Angulo, C. (2014). Real-time video stabilization without phantom movements for micro aerial vehicles. *EURASIP Journal on Image and Video Processing, 1*, 1-13.

Aguilar, W. G., & Angulo, C. (2016). Real-Time Model-Based Video Stabilization for Microaerial Vehicles. *Neural Processing Letters, 43*(2), 459-477.

Aguilar, W. G., & Morales, S. (2016). 3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms. *Electronics, 5*(4), 70.

Aguilar, W. G., Abad, V., Ruiz, H., Aguilar, J., & Aguilar-Castillo, F. (2017). RRT-Based Path Planning for Virtual Bronchoscopy Simulator. En *Lecture Notes in Computer Science* (págs. 155-165).

Aguilar, W. G., Álvarez, L., Grijalva, S., & Rojas, I. (2019). Monocular Vision-Based Dynamic Moving Obstacles Detection and Avoidance. En *Lecture Notes in Computer Science* (págs. 386-398). Germany: Springer.

Aguilar, W. G., Angulo, C., & Costa-Castello, R. (2017). Autonomous Navigation Control for Quadrotors in Trajectories Tracking. En *Lecture Notes in Computer Science* (págs. 287-297).

Aguilar, W. G., Angulo, C., & Pardo, J. A. (2017). Motion intention optimization for multirotor robust video stabilization. *Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON).* Pucón, Chile.

Aguilar, W. G., Angulo, C., Costa, R., & Molina, L. (2014). Control autónomo de cuadricópteros para seguimiento de trayectorias. *IX Congreso de Ciencia y Tecnología ESPE.* Sangolquí, Ecuador.

Aguilar, W. G., Casaliglla, V. P., & Pólit, J. L. (2017). Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles. *Electronics, 6*(1), 10.

Aguilar, W. G., Casaliglla, V. P., Pólit, J. L., Abad, V., & Ruiz, H. (2017). Obstacle Avoidance for Flight Safety on Unmanned Aerial Vehicles. *IWANN, 17(2)*, 195–197.

Aguilar, W. G., Cobeña, B., Rodriguez, G., Salcedo, V. S., & Collaguazo, B. (2018). SVM and RGB-D Sensor Based Gesture Recognition for UAV Control. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (págs. 713-719). Springer.

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Ruiz, H., Parra, H., & Angulo, C. (2017). Pedestrian Detection for UAVs Using Cascade Classifiers and Saliency Maps. En *Lecture Notes in Computer Science* (págs. 563-574).

Aguilar, W. G., Luna, M. A., Moya, J. F., Luna, M. P., Abad, V., Ruiz, H., & Parra, H. (2017). Real-Time Detection and Simulation of Abnormal Crowd Behavior. En *Lecture Notes in Computer Science* (págs. 420-428).

Aguilar, W. G., Luna, M. A., Ruiz, H., Moya, J. F., Luna, M. P., Abad, V., & Parra, H. (2017). Statistical Abnormal Crowd Behavior Detection and Simulation for Real-Time Applications. En *Lecture Notes in Computer Science* (págs. 671-682).

Aguilar, W. G., Luna, M., Moya, J., & Abad, V. (Mayo de 2017). Pedestrian Detection for UAVs Using Cascade Classifiers and Saliency Maps. *International Work-Conference on Artificial Neural Networks*.

Aguilar, W. G., Manosalvas, J. F., Guillén, J. A., & Collaguazo, B. (2018). Robust Motion Estimation Based on Multiple Monocular Camera for Indoor Autonomous

Navigation of Micro Aerial Vehicle. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (págs. 547-561). Springer.

Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017). RRT* GL Based Path Planning for Virtual Aerial Navigation. En *Lecture Notes in Computer Science* (págs. 176-184).

Aguilar, W. G., Quisaguano, F. J., Alvarez, L. G., Pardo, J. A., & Proaño, Z. (2018). Monocular Depth Perception on a Micro-UAV Using Convolutional Neuronal Networks. *Ubiquitous Networking, 10542*, 392–397.

Aguilar, W. G., Quisaguano, F., Rodríguez, G., Alvarez, L., Limaico, A., & Sandoval, D. (2018). Convolutional Neuronal Networks Based Monocular Object Detection and Depth Perception for Micro UAVs. *Lecture Notes in Computer Science*, 401-410.

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017). On-Board Visual SLAM on a UGV Using a RGB-D Camera. En *Lecture Notes in Computer Science* (págs. 298-308).

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017). Real-Time 3D Modeling with a RGB-D Camera and On-Board Processing. En *Lecture Notes in Computer Science* (págs. 410-419).

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017). Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing. *Springer, Cham*, 596–606.

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017). Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing. En *Lecture Notes in Computer Science* (págs. 596-606).

Aguilar, W. G., Salcedo, V., Sandoval, D., & Cobeña, B. (2017). Developing of a Video-Based Model for UAV Autonomous Navigation. *Communications in Computer and Information Science*, 94-105.

Aguilar, W. G., Salcedo, V., Sandoval, D., & Cobeña, B. (2017). Developing of a Video-Based Model for UAV Autonomous Navigation. (C. Springer, Ed.) *Computational Neuroscience. LAWCN 2017. Communications in Computer and Information Science, 720*, 94-105.

Aguilar, W. G., Sandoval, S., Limaico, A., Villegas-Pico, M., & Asimbaya, I. (2019). Path Planning Based Navigation Using LIDAR for an Ackerman Unmanned Ground Vehicle. En *Lecture Notes in Computer Science* (págs. 399-410). Germany: Springer.

Aguilar, W., Angulo, C., Costa, R., & Molina, L. (May de 2014). Control Autónomo de Cuadricopteros para Seguimiento de Trayectorias. *IX CONGRESO DE CIENCIA Y TECNOLOGÍA ESPE 2014*, 144-149.

Albornoz, M., & Calahorrano, D. (2016). *Seguimiento de objetos basado en visión artificial para cuadrirrotor Parrot AR.Drone 2.0.* Tesis Pregrado, Universidad Politécnica Nacional.

Alvarez, L. (2018). *Desarrollo de un sistema vinculado a un Micro-UAV para la detección y evasión de objetos dinámicos a partir de imágenes monoculares.* Tesis Pregrado, Universidad de las Fuerzas Armadas ESPE.

Amaguaña, F., Collaguazo, B., Tituaña, J., & Aguilar, W. G. (2018). Simulation System Based on Augmented Reality for Optimization of Training Tactics on Military Operations. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (págs. 394-403). Springer.

Andrea, C. C., Byron, J. Q., Jorge, P. I., Inti, T. C., & Aguilar, W. G. (2018). Geolocation and Counting of People with Aerial Thermal Imaging for Rescue Purposes. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (págs. 171-182). Springer.

Badrloo, Samira, & Varshosaz, M. (2017). VISION BASED OBSTACLE DETECTION IN UAV IMAGING. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-2/W6*, 21–25.

Basantes, J., Godoy, L., Carvajal, T., Castro, R., Toulkeridis, T., Fuertes, W., . . . Addison, A. (2018). Capture and processing of geospatial data with laser scanner system for 3D modeling and virtual reality of Amazonian Caves. *IEEE Ecuador Technical Chapters Meeting (ETCM).* Samborondón, Ecuador.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *CoRR Computer Vision and Pattern Recognition, abs/2004.10934*, 1-17.

Cabras, P., Rosell, J., Pérez, A., Aguilar, W. G., & Rosell, A. (2011). Haptic-based navigation for the virtual bronchoscopy. *18th IFAC World Congress.* Milano, Italy.

Calderón, M., Aguilar, W., & Merizalde, D. (Mayo de 2020). Visual-Based Real-Time Detection Using Neural Networks and Micro-UAVs for Military Operations. *Developments and Advances in Defense and Security*, 55-64. doi:10.1007/978-981-15-4875-8_5

Colomina, I., & Molina, P. (Junio de 2014). Unmanned aerial systems for photogrammetry and remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing, 92*, 79-97.

Corke, P. (2011). Robotics, vision and control fundamental algorithms in MATLAB. *Springer*.

Cuchango, E., Helbert, E., Sofrony, E., & Jorge, I. (2012). Algoritmo para planear trayectorias de robots móviles, empleando campos potenciales y enjambres de partículas activas brownianas. *Ciencia e Ingeniería Neogranadina, 22*(2), 75–96.

Devos, A., Ebeid, E., & P, M. (2018). Development of Autonomous Drones for Adaptive Obstacle Avoidance in Real World Environments. *Euromicro Conference on Digital System Design*, (págs. 707–710).

Gageik, N., Benz, P., & Montenegro, S. (2015). Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors. *IEEE Access, 3*, 599–609.

Galindo, R., Aguilar, W. G., & Reyes Ch, R. P. (2019). Landmark based eye ratio estimation for driver fatigue detection. En *Lecture Notes in Computer Science* (págs. 565-576). Germany: Springer.

Garcia, J. (2016). *Sistema avanzado de detección de obstáculos y navegación autónoma para vigilancia y protección basado en flota de vehículos aéreos no tripulados.* Tesis Doctoral, Universidad de Málaga., Málaga.

Garcia, S., López, M., Barea, R., Bergasa, L., Gómez, A., & Molinos, E. (2016). Indoor SLAM for Micro Aerial Vehicles Control Using Monocular Camera and Sensor Fusion. *International Conference on Autonomous Robot Systems and Competitions (ICARSC).*

Godard, C., Aodha, O., & Brostow, G. (April de 2017). Unsupervised Monocular Depth Estimation with Left-Right Consistency. *2017 IEEE Conference On Computer Vision And Pattern Recognition (CVPR)*.

Grijalva, S. (2019). *Sistema de navegación autónomo basado en visión para pistas de carreras de drones con marcas de superficie.* Tesis Pregrado, Universidad de las Fuerzas Armadas ESPE, Sangolquí.

Grijalva, S., & Aguilar, W. G. (2019). Landmark-Based Virtual Path Estimation for Assisted UAV FPV Tele-Operation with Augmented Reality. En *Lecture Notes in Computer Science* (págs. 688-700). Germany: Springer.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., . . . Chen, T. (2018, Mayo). Recent Advances in Convolutional Neural Networks. *Pattern Recognition, 77*, 354-377.

Hyun, L., Apvrille, T. T., & Dugelay, J.-L. (2014). Autonomous drones for assisting rescue services within the context of natural disasters. *General Assembly and Scientific Symposium (URSI GASS), 2014 XXXIth URSI. IEEE*, 1–4.

Jain, S. (2018). *Nanonets*. Recuperado el 15 de Julio de 2020, de How to easily Detect Objects with Deep Learning on Raspberry Pi: https://nanonets.com/blog/how-to-easily-detect-objects-with-deep-learning-on-raspberry-pi/?&utm_source=nanonets.com/blog/&utm_medium=blog&utm_content=How%20to%20add%20Person%20Tracking%20to%20a%20Drone%20using%20Deep%20Learning%20and%20NanoNets

Jara-Olmedo, A., Medina-Pazmiño, W., Mesías, R., Araujo-Villaroel, B., Aguilar, W. G., & Pardo, J. A. (2018). Interface of Optimal Electro-Optical/Infrared for Unmanned Aerial Vehicles. En *Smart Innovation, Systems and Technologies* (págs. 372-380).

Jara-Olmedo, A., Medina-Pazmiño, W., Tozer, T., Aguilar, W. G., & Pardo, J. A. (2018). E-services from Emergency Communication Network: Aerial Platform Evaluation. *International Conference on eDemocracy & eGovernment (ICEDEG)* (págs. 251-256). IEEE.

Khademi, Gholamreza; Simon, Dan. (2019). Convolutional Neural Network for Environmentally Aware Locomotion Mode Recognition of Lower-Limb Amputees. *ASME Dynamic Systems and Control Conference & DSCC2019.* Park City, Utah.

Khan, G., Tariq, Z., & Ghani Khan, M. (2019). Multi-Person Tracking Based on Faster R-CNN and Deep Appearance Features. En P. Mazzeo, S. Ramakrisnan, & P. Spagnolo, *Visual Object Tracking with Deep Neural Networks.* Londres: IntechOpen. doi:10.5772/intechopen.85215

Kleiven, K., Liu, L., McVeety, R., & Liu, R. (2012). Robot Hide and Seek.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional. *Advances in Neural Information Processing Systems 25 (NIPS 2012)*.

Lacey, T. (2015). *MIT - Massachusetts Institute of Technology.* Recuperado el 1 de Junio de 2020, de Tutorial: The Kalman Filter: http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf

Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., & Navab, N. (2016). Deeper Depth Prediction with Fully Convolutional Residual Network. *Published at IEEE International Conference on 3D Vision (3DV).*

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (Noviembre de 1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324. doi:10.1109/5.726791

López, M., García, S., Barea, R., Bergasa, L., Molinos, E., Arroyo, R., . . . Pardo, S. (2017, Abril). A Multi-Sensorial Simultaneous Localization andMapping (SLAM) System for Low-Cost Micro AerialVehicles in GPS-Denied Environments. *Sensors 2017, 17*(4).

Mark Robins. (27 de Mayo de 2020). *The Difference Between Artificial Intelligence, Machine Learning and Deep Learning*. Obtenido de Intel.com: https://www.intel.com/content/www/us/en/artificial-intelligence/posts/difference-between-ai-machine-learning-deep-learning.html

Menon, N., & Somawat, V. (8 de Octubre de 2018). *GitHub.* Recuperado el 15 de Noviembre de 2019, de ambakick/Person-Detection-and-Tracking: https://github.com/ambakick/Person-Detection-and-Tracking

missinglink.ai. (s.f.). *Image Segmentation in Deep Learning: Methods and Applications*. Recuperado el 9 de Junio de 2020, de missinglink.ai: https://missinglink.ai/guides/neural-network-concepts/perceptrons-and-multi-layer-perceptrons-the-artificial-neuron-at-the-core-of-deep-learning/

Mora, M., & Tornero, J. (2007). Planificación de movimientos mediante la propagación de campos potenciales artificiales. *8° Congreso Iberoamericano de Ingeniería Mecánica.*

Morales, L., & Paucar, C. (2017). *Investigación e implementación de un sistema de seguridad fijo y móvil mediante un dron, usando visión artificial para detección y seguimiento de personas en un ambiente externo específico, de la Universidad de las Fuerzas Armadas ESPE-L.* Tesis Pregrado, Latacunga.

Mur-Artal, R., & Tards, J. D. (2017). Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters, 2*(2), 796–803.

Nielsen, M. (Diciembre de 2019). *Deep Learning*. Recuperado el 7 de Enero de 2020, de Neural Networks and Deep Learning: http://neuralnetworksanddeeplearning.com/chap6.html

Orbea, D., Moposita, J., Aguilar, W. G., Paredes, M., León, G., & Jara-Olmedo, A. (2017). Math Model of UAV Multi Rotor Prototype with Fixed Wing Aerodynamic Structure for a Flight Simulator. En *Lecture Notes in Computer Science* (págs. 199-211).

Orbea, D., Moposita, J., Aguilar, W. G., Paredes, M., Reyes, R. P., & Montoya, L. (2017). Vertical take off and landing with fixed rotor. *Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON).* Pucón, Chile.

paperswithcode. (2020). *Object Detection on COCO test-dev*. Recuperado el 12 de Febrero de 2020, de paperswithcode.com: https://paperswithcode.com/sota/object-detection-on-coco

Pardo, J. A., Aguilar, W. G., & Toulkeridis, T. (2017). Wireless communication system for the transmission of thermal images from a UAV. *Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON).* Pucón, Chile.

Parrot SA. (2020). *Parrot Store Oficial*. Recuperado el 5 de Mayo de 2020, de Parrot Bebop 2 FPV drone - Technical Specifications: https://www.parrot.com/us/drones/parrot-bebop-2-fpv#

Quisaguano, F. (2018). *Desarrollo de un sistema simultáneo de seguimiento de persona y evasión de obstáculos no definidos usando estimación de profundidad para imágenes monoculares en un MICRO-UAV.* Tesis Pregrado, Universidad de las Fuerzas Armadas ESPE.

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *Computer Science - Computer Vision and Pattern Recognition*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., . . . Li, F. (2015, Abril 11). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision, 115*, 211-252. doi:https://doi.org/10.1007/s11263-015-0816-y

Salcedo, V. (2018). *Aterrizaje automático de un vehículo aéreo no tripulado basado en seguimiento de puntos de interés para superficies móviles.* Tesis Pregrado, Universidad de las Fuerzas Armadas ESPE.

Shirzadeh M., a. e. (2015). An indirect adaptive neural control of a visual-based quadrotor robot for pursuing a moving target. *ISA Transactions*.

Shirzadeh, M. (2017). Vision-based control of a quadrotor utilizing artificial neural networks for tracking of moving targets. *Engineering Applications of Artificial Intelligence, 58*, 34–48.

Stanford. (Junio de 2020). *CS231n: Convolutional Neural Networks for Visual Recognition*. Recuperado el 23 de Febrero de 2020, de Stanford.edu: https://cs231n.github.io/convolutional-networks/

Sturm, P., Ramalingam, S., Tardif, J., Gasparini, S., & Barreto, J. (2011). Camera Models and Fundamental Concepts Used in Geometric Computer Vision. *Foundations and Trends in Computer Graphics and Vision, 6*(1-2), 1-183.

Sumimoto, S., Miyata, Y., Yoshimura, R., Uwate, Y., & Nishio, Y. (2018). Design of Convolutional Neural Network for Classifying Depth Prediction Images from Overhead. *International SoC Design Conference (ISOCC)*, (págs. 166–167).

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (págs. 1701-1708).

TensorFlow.org. (2019). *Learn Tensorflow 4: Convolutional Neural Networks (CNNs)*. Recuperado el 15 de Marzo de 2020, de TensorFlow Core: https://codelabs.developers.google.com/codelabs/tensorflow-lab4-cnns/#0

Valencia, D., & Kim, D. (2018). Quadrotor Obstacle Detection and Avoidance System Using a Monocular Camera. *3rd Asia-Pacific Conference on Intelligent Robot Systems*, (págs. 78–81).

Zulu, A., & John, S. (December de 2014). A Review of Control Algorithms for Autonomoues Quadrotors. *Open Journal of Applied Sciences, 4*, 547-556.